MICROHOBBY AMSTRAD Especial

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD AÑO I N.º 3

350 ptas.
Canarias, Ceuta y Melilla 335 ptas.

SIMULADORES DE VUELO: !COMPARATIVO MONSTRUO!

ROBOTS: EL FUTURO EMPEZO AYER

EL INCREIBLE MUNDO DE BLOQUES. ¿ES UN PROGRAMA O ERES TU?

EL NO VA MAS DE LOS LENGUAJES, ESCRITO POR NOSOTROS PARA TI. TE PRESENTAMOS A... FORTH

EL FASCINANTE JUEGO DE LA VIDA

HOBBY PRESS

Sound-on-Sound

La cinta virgen para ordenador C15 y C20



Director Editorial
José I. Gómez-Centurión
Director Ejecutivo
José M.ª Díaz
Redactor Jefe
Juan José Martínez
Diseño gráfico
Fernando Chaumel
Colaboradores
Eduardo Ruiz
Javier Barceló
David Sopuerta
Robert Chatwin
Francisco Portalo
Pedro Sudón
Miguel Sepúlveda
Francisco Martín
Jesús Alonso
Pedro S. Pérez
Amalio Gómez
Secretaria Redacción
Carmen Santamaría
Fotografía
Carlos Candel
Portada
M. Barco
Ilustradores
J. Igual, J. Pons, F. L. Frontán, J. Septien, Pejo, J. J. Mora

Edita
HOBBY PRESS, S.A.

Presidente
María Andrino
Consejero Delegado
José I. Gómez-Centurión

Jefe de Producción
Carlos Peropadre

Marketing
Marta García
Jefe de Publicidad
Concha Gutiérrez
Publicidad Barcelona
José Galán Cortés
Tél: (93) 303 10 22/313 71 62

Secretaria de Dirección
Marisa Cogorro

Suscripciones
M.ª Rosa González
M.ª del Mar Calzada

Redacción, Administración y Publicidad
Ctra. de Irún km 12,400
(Fuencarral) 28049 Madrid
Teléfonos: Suscrip.: 734 65 00
Redacción: 734 70 12

Dto. Circulación
Paulino Blanco

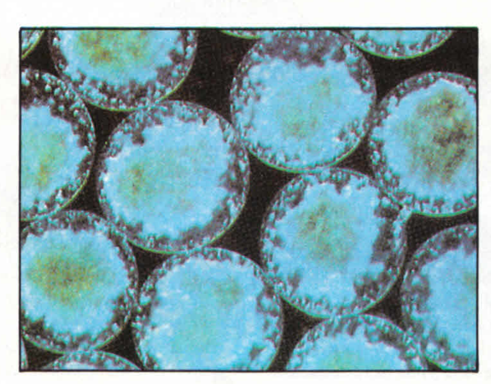
Distribución
Coedis, S. A. Valencia, 245
Barcelona

Imprime
Gráficas Reunidas
Avda. Aragón, 56 (MADRID)
Fotocomposición
Novocomp, S.A.
Nicolás Morales, 38-40
Fotomecánica
GROF
Ezequiel Solana, 16
Depósito Legal:
M-5836-1986

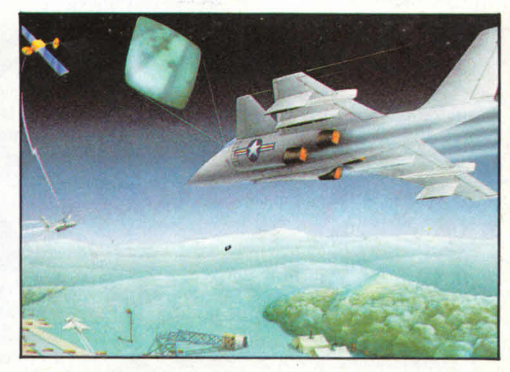
MICROHOBBY AMSTRAD
sumario

Año I • Número 3 • Octubre 1986
Precio 350 ptas. Canarias, Ceuta y Melilla 335 ptas.

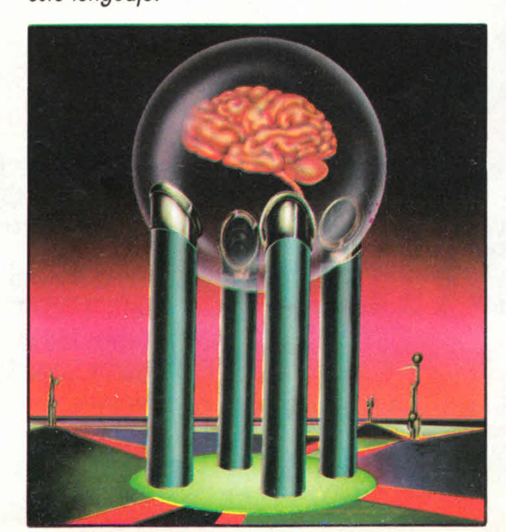
Lo que está vivo, ¿tiene leyes? ¿Cómo evoluciona una forma viviente? Este fascinante juego lo muestra de principio a fin. No es el comienzo de la genética asistida por ordenador, pero se le parece tanto... Además, todo en lenguaje máquina.



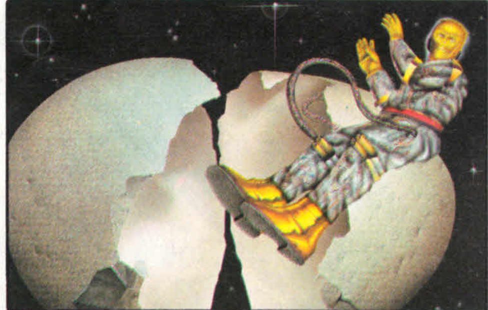
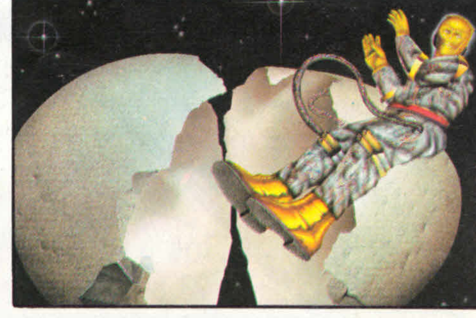
Hacía falta una cosa así. Alguien tenía que probar todos los simuladores de vuelo que existen para ***Amstrad,*** *es decir, los* ***«juegos»*** *más espectaculares para ordenador, hábil y objetivamente juzgada por nuestro equipo de expertos en software.*



El lenguaje Forth es por antonomasia uno de los favoritos de los programadores, usado a menudo como herramienta de desarrollo. Su funcionalidad, junto a su proximidad a la máquina trabajando a un nivel verdaderamente bajo, le capacitan como un verdadero sustitutivo del en ocasiones trabajoso y cansino código máquina. ***AMSTRAD ESPECIAL*** *número 3 os ofrece un intérprete de FORTH, ideal para iniciarse en este lenguaje.*



Los ordenadores están aquí. Sus hijos, los robots, también, y ya están empezando a ejercer una importante influencia en nuestras vidas. La Segunda Revolución Informática empezò ayer.

Hay cosas que son increíbles, como un elefante que vuele, o como este programa, dentro de la más pura tradición de la IA, que ***«existe»*** *en un mundo imaginario regido por una serie de leyes, las cuales le permiten entender castellano normal y ejecutar órdenes de gran complejidad. Simplemente, es increíble.*



DE VENTA EN LOS MEJORES COMERCIOS DE INFORMATICA
Si Vd. tiene alguna dificultad para obtener los programas, puede dirigirse a:
Ofites Informática
Avda. Isabel II, 16 - 8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN
CONDICIONES ESPECIALES PARA DISTRIBUIDORES
EDITOR Y DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA

EL JUEGO DE LA VIDA

Cuando John Horton Conway, famoso matemático de la Universidad de Cambridge, creó por primera vez en su ordenador el conocido juego de la vida, no podía imaginar el revuelo que éste causaría; aun después de tanto tiempo, sería tremendamente difícil evaluar las pérdidas económicas que su juego provocó. En todas las grandes empresas de EE.UU., allí donde existiese un ordenador, miles de seres unicelulares nacían y morían tras un monitor verde.

Ahora, años después, AMSTRAD Especial ofrece a todos sus lectores la posibilidad de recrearse en este universo de fantasía, donde la vida y la muerte se aúnan en una lucha eterna en pos de la simetría.

El juego de la vida, a causa de sus semejanzas con el nacimiento, muerte y alteraciones que experimentan las sociedades de seres vivos, pertenece a la clase de los llamados juegos de simulación.

Todo el proceso del juego encaja con la teoría de autómatas celulares, la cual propone la posibilidad de que una máquina provista de las instrucciones necesarias, puede ser capaz de construir una copia de si misma.

Cada una de estas máquinas sería a su vez capaz de construir otras, y estas cuatro se convertirían en ocho, y así sucesivamente.

Ahora bien, tal proceso conduciría a un número infinito de autómatas, lo cual sería insostenible. Para evitar esta proliferación, Conway propuso la aplicación de unas leyes genéticas que regularan los nacimientos, muertes y sucesivas alteraciones.

EL CICLO DE NACIMIENTO Y MUERTE

Dicho juego debe jugarse teóricamente sobre un tablero cuadriculado infinito. Cada una de estas casillas puede encontrarse en dos estados, vacío o lleno. Cada uno de estas cuadrículas tiene asociadas un conjunto finito de cuadrículas 'vecinas' que pueden tener influencia sobre su estado.

La configuración de estados cambia a intervalos temporales, en conformidad con unas reglas de transición, aplicadas simultánemanete a todas las cuadrículas.

Conway eligió sus reglas, tras un período de experimentación, intentando satisfacer tres condiciones:

1. No debe existir ninguna configuración inicial para la que se pueda demostrar fácilmente que su población crecerá ilimitadamente.

2. Deben existir configuraciones iniciales que aparentemente crezcan sin límite.

3. Han de exisitir configuraciones iniciales sencillas que sean capaces de crecer y cambiar durante períodos de tiempo considerables, antes de finalizar de una de las siguientes formas posibles:

a) Extinguirse completamente, ya sea por superpoblación o por encarecimiento.

b) Adoptar una configuración estable invariable en lo sucesivo.

c) Entrar en fase oscilatoria, donde se repiten sin fin dos o más estados.

LAS LEYES DE CONWAY

Las leyes genéticas elegidas por Conway, que son sobre las que está basado el programa que os ofrecemos del **'Juego de la vida'**, son las siguientes:

1. Supervivencia. Cada ficha que tenga dos o tres fichas vecinas sobrevive y pasa a la siguiente generación.

2. Fallecimiento. Cada ficha que tenga cuatro o más vecinas, muere por superpoblación. Las fichas con sólo una o ninguna vecinas muere por aislamiento.

3. Nacimientos. Cada casilla vacía adyacente a exactamente tres fichas vecinas, es casilla generatriz. Por lo que deberá colocarse allí una ficha.

Debemos hacer notar que cada ficha posee ocho vecinas, es decir, cuatro ortogonalmente y otras cuatro diagonalmente.

Es importante darse cuenta que todos los nacimientos y muertes ocurren simultáneamente, y constituyen en su conjunto una generación, o como se las suele llamar un **'tic'** o **'latido'** de la vida de la configuaración inicial.

Descubriremos, una vez iniciado el juego, que la población experimenta constantemente cambios insólitos, bellos e inesperados. En ciertos casos, la sociedad termina por extinguirse, si bien antes de que esto suceda, deberán pasar gran número de generaciones.

Casi todas las configuraciones iniciales terminan por alcanzar figuras estables (que Conway llama naturalezas muertas), incapaces de cambio y que quedan oscilando.

Las formaciones iniciales que no poseen simetría, tienden a ir adquiriéndola, y una vez que esto sucede ya no puede perderse.

Una familia formada únicamente por dos fichas, se extinguirán al primer latido.

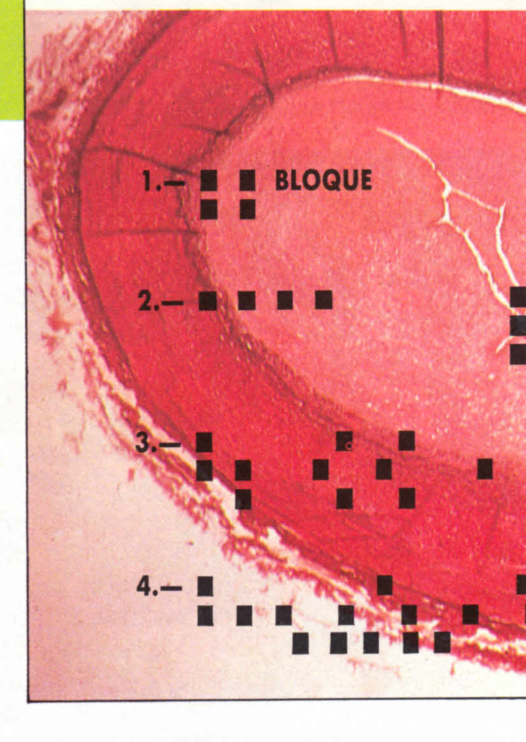
Una formación inicial de tres casillas moriría de inmediato a menos que una de ellas tenga un par de vecinas.

Debemos hacer notar que una cadena diagonal aislada de fichas, por larga que ésta sea, termina desapareciendo, ya que en cada latido pierde las dos fichas situadas en sus extremos.

La tercera configuración se transforma en un 'BLOQUE' estable en el segundo latido.

La última de ellas es la más sencilla de las llamadas **'FLIP-FLOPS'** que son figuras oscilantes de período dos. Conway la denomina **'intermitente'**.

Veamos ahora algunas figuras compuestas por cuatro fichas conectadas entre sí por movimientos de torre, denominadas tetróminos:



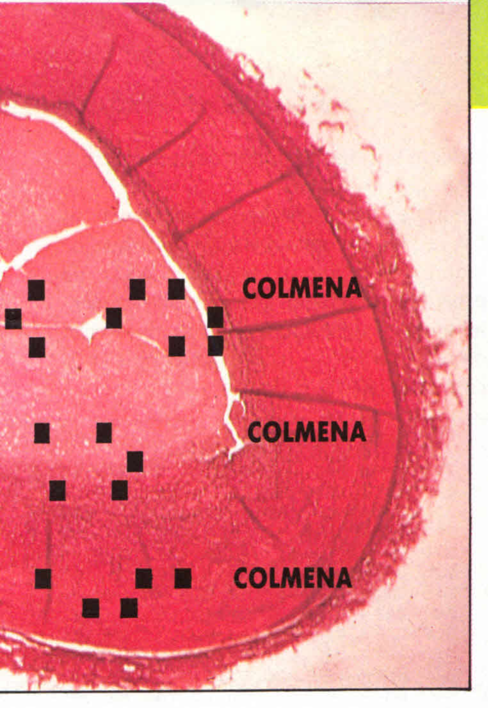
El primero de ellos es como hemos visto anteriormente, una naturaleza muerta.

Los tetróminos 2 y 3, alcanzan una configuración estable al segundo latido, dicha configuración estable, se denomina **«colmena».** El último tetrómino se convierte en colmena al tercer latido.

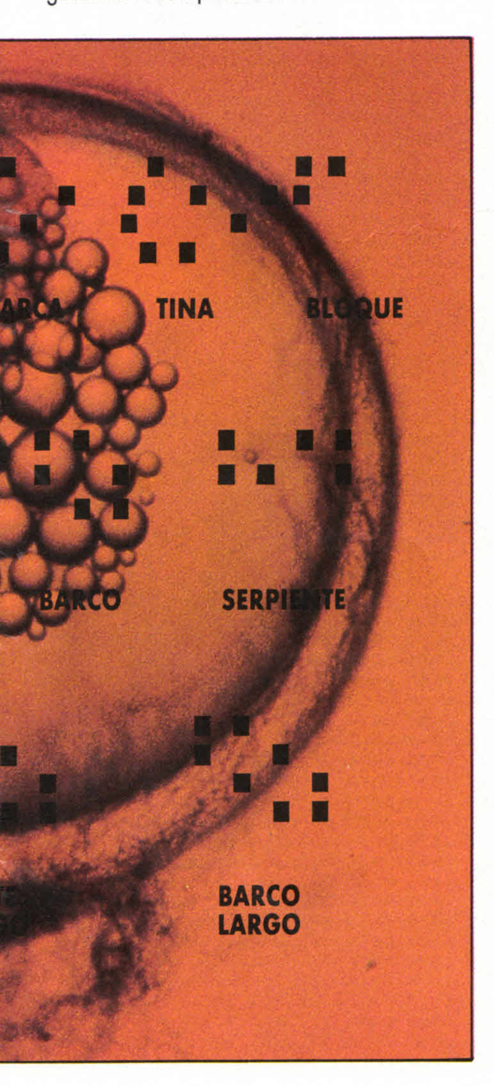
Vamos a ver a continuación algunas de las



gráfico n.º 1



figuras más interesantes que se pueden presentar a través de las diferentes fases del juego de la vida. En el primer gráfico podemos observar algunas de las formas estables más corrientes. Dichas formas son las fases iniciales de muchas configuraciones, y no varían a lo largo del tiempo, a no ser que otras configuraciones se aproximen a ellas.



Una de las figuras más interesantes con que nos podemos encontrar, es el llamado deslizador», formado por cinco casillas, y que podemos ver en el gráfico 2.

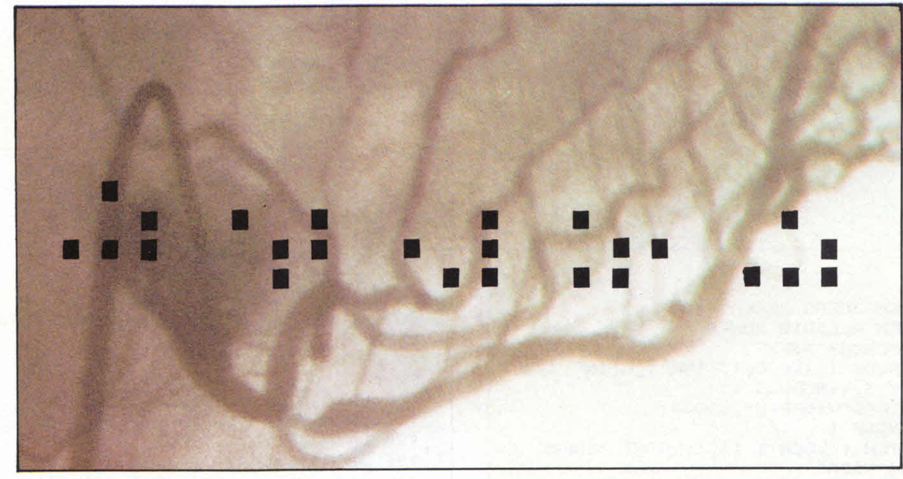


gráfico n.° 2

Dicho deslizador, al cabo de dos latidos, se desplaza y queda reflejado respecto de una recta diagonal. Al cabo de otros dos latidos, se vuelve a enderezar, desplazándose una cuadrícula digonalmente hacia abajo y hacia la derecha de su posición inicial.

Una vez explicadas las curiosidades más interesantes del juego de la vida, pasaremos ahora a ver cómo funciona el programa que aparece al final del artículo.

EL PROGRAMA

El programa consta de un bloque en Basic y otro en código máquina. El primero de ellos se encarga de imprimir en pantalla el menú de opciones que posee el juego de la vida, así como todas las teclas que pueden ser utilizadas en cada parte del programa.

Las tres opciones disponibles, son las siguientes: Creación de pantalla, Pulsaciones y Pantalla aleatoria.

La primera de ellas nos ofrece la posibilidad de crear nuestra propia pantalla utilizando las teclas de cursor para desplazarnos por la pantalla.

Pulsando la tecla **«P»** podremos pintar utilizando los cursores y pulsando la tecla **«O»** podremos borrar o desplazarnos por la pantalla sin pintar.

Una vez hayamos finalizado nuestra pantalla bastará con pulsar la tecla **«COPIA»** para que empiecen a producirse los latidos.

Cuando estemos en la fase de pulsaciones, tendremos las opciones de parar las pulsaciones, pulsando la tecla «P» o bien retornar el menú principal pulsando la tecla «B».

Cuando nos encontremos en **«PAUSA»**, podremos volver a iniciar los latidos pulsando la tecla **«E»** o bien salvar en disco o cinta la pantalla actual, pulsando la tecla «S».

La segunda de las opciones con la que nos encontramos, es la de entrar directamente a la fase de pulsaciones. Lógicamente si utilizamos esta opción cuando no tengamos ninguna pantalla en memoria, no se producirá ningún efecto. Así pues, esta opción servirá cuando estando en la fase de pulsaciones se haya vuelto al menú y se desee retornar a las pulsaciones.

Por último, tenemos la opción de crear una pantalla aleatoria. Eligiendo esta opción, nuestro **Amstrad** creará una pantalla al azar y después de esto pasará directamente a la fase de pulsaciones.

Veamos ahora cuáles son los pasos que realiza el programa en código máquina.

En primer lugar, se crea un buffer de 1.200 bytes, sobre el cual se producen todos los cálculos para comprobar cuáles son las células que sobreviven, nacen o mueren.

Dentro de ese buffer de trabajo, cada bit contiene la información de una casilla, y puede tener dos estados: 1 indica que la célula está viva y 0 indicará que dicha célula está muerta.

Así pues, en cada latido se deben comprobar cada uno de los bits que componen ese buffer de 1.200 bytes, por lo tanto deberemos tratar 1.200*8 bits.

Como hemos dicho anteriormente las reglas sobre las que se basa el juego contemplan las ocho casillas inmediatamente próximas a la que se está estudiando.

De este modo, cuando deseemos mirar lo que ocurriría con un bit en concreto, deberemos mirar cada uno de los ocho bits que lo rodean.

Si por ejemplo deseamos mirar el bit 4 de un byte cualquiera dentro del buffer, deberemos mirar los siguientes bits de los bytes que se indican:

```
a b c
d x e
f g h
```

Así pues, teniendo en cuenta que el número de bytes que contiene cada fila del buffer

es 10 y si cargamos en el registro indexado IX la posición de memoria en la cual está el bit que intentamos investigar, los bits que deberemos chequear serán los siguientes:

a. bit 3 de (IX-10)
b. bit 4 de (IX-10)
c. bit 5 de (IX-10)
d. bit 3 de (IX+0)
e. bit 5 de (IX+0)
f. bit 3 de (IX+10)
g. bit 4 de (IX+10)
h. bit 5 de (IX+10)

De esta forma cuando uno de estos bits esté puesto a uno, llamamos a una rutina que se encarga de incrementar el contador que nos indicará finalmente cuántos de esos ocho bits están puestos a uno.

Así pues, una vez revisados cada uno de los bits vecinos, tomaremos el valor de dicho contador, y de esta forma se podrá decidir, de acuerdo con las reglas indicadas anteriormente, si ese bit debe sobrevivir o morir en caso de que esté vivo, o bien si debe nacer en el caso de que esté muerto.

```
1 REM JUEGO DE LA VIDA
2 REM ALBERTO SUNER
10 MEMORY &3FFF
20 MODE 1:INK 0,13:INK 1,0:INK 2,20
:INK 3,1:BORDER 13
30 LOAD"VIDABIN",&A000
40 MODE 1
50 PEN 1:LOCATE 13,1:PRINT "JUEGO D
E LA VIDA"
60 PEN 2:LOCATE 5,2:PRINT "MENU":PE
N 3
70 LOCATE 5,4:PRINT "1.............
 CREACION PANTALLA"
80 LOCATE 5,5:PRINT "2.............
 PULSACIONES"
90 LOCATE 5,6:PRINT "3.............
 PANTALLA ALEATORIA"
100 PEN 2:LOCATE 5,8:PRINT "CREACIO
N DE PANTALLA"
110 PEN 3:LOCATE 5,10:PRINT "CURSOR
ES...... MUEVEN PUNTO"
120 LOCATE 5,11:PRINT "COPIA.......
.. LATIDOS"
130 LOCATE 5,12:PRINT "P..........
.. PINTAR"
140 LOCATE 5,13:PRINT "O..........
.. BORRAR"
150 LOCATE 5,14:PRINT "L..........
.. CARGA PANTALLA"
160 LOCATE 5,15:PRINT "B..........
.. MENU"
170 PEN 2:LOCATE 5,17:PRINT "LATIDO
S":PEN 3
180 LOCATE 5,19:PRINT "P..........
.. PAUSA"
190 LOCATE 5,20:PRINT "B..........
.. MENU"
200 PEN 2:LOCATE 5,22:PRINT "PAUSA"
:PEN 3
210 LOCATE 5,24:PRINT "S..........
.. SALVA PANTALLA"
220 LOCATE 5,25:PRINT "E..........
.. SEGUIR CON LATIDOS"
230 IF INKEY(64)=0 THEN 350
240 IF INKEY(65)=0 THEN 390
250 IF INKEY(57)=0 THEN 270
260 GOTO 230
270 MODE 2
280 FOR N=&9000 TO &9200
290 POKE N,INT(RND*256)
300 NEXT
310 POKE &A553,0:POKE &A554,0
320 GOSUB 430
330 CALL &A134
340 GOTO 40
350 MODE 2
360 GOSUB 430
370 CALL &A000
380 GOTO 40
390 MODE 2
400 GOSUB 430
410 CALL &A134
420 GOTO 40
430 LOCATE 30,25:PRINT "LATIDOS-000
00"
440 RETURN
```

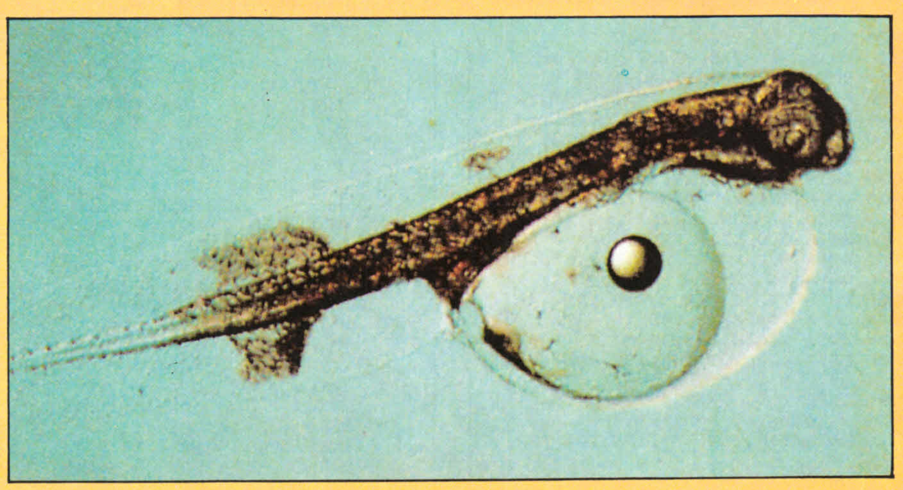
```
10 REM JUEGO DE LA VIDA
20 REM PROGRAMA CARGADOR
30 FOR N=&A000 TO &A5A3
40 READ A:SUMA=SUMA+A
50 POKE N,A
60 NEXT
70 IF SUMA<>173577 THEN PRINT "ERRO
R EN DATAS"
80 DATA 33,0,0,34,83,165,33
90 DATA 0,144,17,1,144,1,176
100 DATA 4,54,0,237,176,33,165
110 DATA 144,34,6,165,62,54,205
120 DATA 30,187,192,62,36,205,30
130 DATA 187,40,3,205,58,165,175
140 DATA 205,30,187,40,3,205,238
150 DATA 160,62,2,205,30,187,40
160 DATA 3,205,252,160,62,8,205
170 DATA 30,187,40,3,205,7,161
180 DATA 62,1,205,30,187,40,3
190 DATA 205,29,161,62,27,205,30
200 DATA 187,40,4,175,50,9,165
210 DATA 62,34,205,30,187,40,5
220 DATA 62,1,50,9,165,205,116
230 DATA 160,62,9,205,30,187,194
240 DATA 52,161,24,165,42,6,165
250 DATA 58,8,165,254,0,32,2
260 DATA 203,198,254,1,32,2,203
270 DATA 206,254,2,32,2,203,214
280 DATA 254,3,32,2,203,222,254
290 DATA 4,32,2,203,230,254,5
300 DATA 32,2,203,238,254,6,32
310 DATA 2,203,246,254,7,32,2
320 DATA 203,254,205,57,164,58,9
330 DATA 165,167,32,1,201,42,6
340 DATA 165,58,8,165,254,0,32
350 DATA 2,203,134,254,1,32,2
360 DATA 203,142,254,2,32,2,203
370 DATA 150,254,3,32,2,203,158
380 DATA 254,4,32,2,203,166,254
390 DATA 5,32,2,203,174,254,6
400 DATA 32,2,203,182,254,7,32
410 DATA 2,203,190,205,57,164,201
420 DATA 42,6,165,17,10,0,55
430 DATA 63,237,82,34,6,165,201
440 DATA 42,6,165,17,10,0,25
450 DATA 34,6,165,201,58,8,165
460 DATA 60,50,8,165,254,8,192
470 DATA 42,6,165,43,34,6,165
480 DATA 175,50,8,165,201,58,8
490 DATA 165,61,50,8,165,254,255
500 DATA 192,42,6,165,35,34,6
510 DATA 165,62,7,50,8,165,201
520 DATA 33,0,149,17,1,149,1
530 DATA 88,2,54,0,237,176,33
540 DATA 0,149,221,33,0,144,1
550 DATA 88,2,221,94,0,22,0
560 DATA 203,123,40,11,205,62,162
570 DATA 205,14,164,204,22,164,24
580 DATA 9,205,62,162,205,9,164
590 DATA 204,22,164,22,0,203,115
600 DATA 40,11,205,119,162,205,14
610 DATA 164,204,25,164,24,9,205
620 DATA 119,162,205,9,164,204,25
630 DATA 164,22,0,203,107,40,11
640 DATA 205,176,162,205,14,164,204
650 DATA 28,164,24,9,205,176,162
660 DATA 205,9,164,204,28,164,22
670 DATA 0,203,99,40,11,205,233
680 DATA 162.205.14.164.204,31,164
690 DATA 24,9,205,233,162,205,9
700 DATA 164,204,31,164,22,0,203
710 DATA 91,40,11,205,34,163,205
720 DATA 14,164,204,34,164,24,9
730 DATA 205,34,163,205,9,164,204
740 DATA 34,164,22,0,203,83,40
750 DATA 11,205,91,163,205,14,164
760 DATA 204,37,164,24,9,205,91
770 DATA 163,205,9,164,204,37,164
780 DATA 22,0,203,75,40,11,205
790 DATA 148,163,205,14,164,204,40
800 DATA 164,24,9,205,148,163,205
810 DATA 9,164,204,40,164,22,0
820 DATA 203,67,40,11,205,205,163
830 DATA 205,14,164,204,43,164,24
840 DATA 9,205,205,163,205,9,164
850 DATA 204,43,164,221,35,35,11
860 DATA 120,177,194,75,161,205,46
870 DATA 164,205,85,165,62,27,205
880 DATA 30,187,196,10,165,62,54
890 DATA 205,30,187,202,52,161,201
900 DATA 221,203,255,70,196,7,164
910 DATA 221,203,0,118,196,7,164
920 DATA 221,203,9,70,196,7,164
930 DATA 221,203,10,126,196,7,164
940 DATA 221,203,10,118,196,7,164
950 DATA 221,203,246,118,196,7,164
960 DATA 221,203,246,126,196,7,164
970 DATA 221,203,245,70,196,7,164
980 DATA 201,221,203,0,126,196,7
990 DATA 164,221,203,0,110,196,7
1000 DATA 164,221,203,246,126,196,7
1010 DATA 164,221,203,246,118,196,7
1020 DATA 164,221,203,246,110,196,7
1030 DATA 164,221,203,10,126,196,7
1040 DATA 164,221,203,10,118,196,7
1050 DATA 164,221,203,10,110,196,7
1060 DATA 164,201,221,203,0,118,196
1070 DATA 7,164,221,203,0,102,196
1080 DATA 7,164,221,203,246,118,196
1090 DATA 7,164,221,203,246,110,196
1100 DATA 7,164,221,203,246,102,196
1110 DATA 7,164,221,203,10,118,196
1120 DATA 7,164,221,203,10,110,196
1130 DATA 7,164,221,203,10,102,196
1140 DATA 7,164,201,221,203,0,110
1150 DATA 196,7,164,221,203,0,94
1160 DATA 196,7,164,221,203,246,110
1170 DATA 196,7,164,221,203,246,102
1180 DATA 196,7,164,221,203,246,94
1190 DATA 196,7,164,221,203,10,110
1200 DATA 196,7,164,221,203,10,102
1210 DATA 196,7,164,221,203,10,94
1220 DATA 196,7,164,201,221,203,0
1230 DATA 102,196,7,164,221,203,0
1240 DATA 86,196,7,164,221,203,246
1250 DATA 102,196,7,164,221,203,246
1260 DATA 94,196,7,164,221,203,246
1270 DATA 86,196,7,164,221,203,10
1280 DATA 102,196,7,164,221,203,10
1290 DATA 94,196,7,164,221,203,10
1300 DATA 86,196,7,164,201,221,203
1310 DATA 0,94,196,7,164,221,203
1320 DATA 0,78,196,7,164,221,203
1330 DATA 246,94,196,7,164,221,203
1340 DATA 246,86,196,7,164,221,203
1350 DATA 246,78,196,7,164,221,203
1360 DATA 10,94,196,7,164,221,203
1370 DATA 10,86,196,7,164,221,203
1380 DATA 10,78,196,7,164,201,221
1390 DATA 203,0,86,196,7,164,221
1400 DATA 203,0,70,196,7,164,221
1410 DATA 203,246,86,196,7,164,221
1420 DATA 203,246,78,196,7,164,221
1430 DATA 203,246,70,196,7,164,221
1440 DATA 203,10,86,196,7,164,221
1450 DATA 203,10,78,196,7,164,221
1460 DATA 203,10,70,196,7,164,201
1470 DATA 221,203,0,78,196,7,164
1480 DATA 221,203,1,126,196,7,164
1490 DATA 221,203,246,78,196,7,164
1500 DATA 221,203,246,70,196,7,164
1510 DATA 221,203,247,126,196,7,164
1520 DATA 221,203,10,78,196,7,164
1530 DATA 221,203,10,70,196,7,164
```

```
1540 DATA 221,203,11,126,196,7,164
1550 DATA 201,201,20,201,122,254,3
1560 DATA 200,201,122,254,2,200,254
1570 DATA 3,200,201,203,254,201,203
1580 DATA 246,201,203,238,201,203,2
30
1590 DATA 201,203,222,201,203,214,2
01
1600 DATA 203,206,201,203,198,201,3
3
1610 DATA 0,149,17,0,144,1,88
1620 DATA 2,237,176,33,0,64,17
1630 DATA 1,64,1,0,64,54,0
1640 DATA 237,176,205,77,164,205,19
5
1650 DATA 164,201,17,0,64,33,0
1660 DATA 144,6,60,197,237,83,182
1670 DATA 164,6,10,175,203,126,196
1680 DATA 184,164,18,175,19,203,118
1690 DATA 196,184,164,18,175,19,203
1700 DATA 110,196,184,164,18,175,19
1710 DATA 203,102,196,184,164,18,17
5
1720 DATA 19,203,94,196,184,164,18
1730 DATA 175,19,203,86,196,184,164
1740 DATA 18,175,19,203,78,196,184
1750 DATA 164,18,175,19,203,70,196
1760 DATA 184,164,18,19,35,16,189
1770 DATA 229,42,182,164,1,80,0
1780 DATA 237,176,33,80,0,25,235
1790 DATA 225,193,16,164,237,83,182
1800 DATA 164,201,0,0,203,247,203
1810 DATA 255,203,239,203,231,203,2
23
1820 DATA 201,33,1,1,17,0,64
1830 DATA 213,235,33,176,191,66,22
1840 DATA 0,29,25,17,80,0,25
1850 DATA 16,253,221,225,6,192,24
1860 DATA 21,124,230,56,254,56,40
1870 DATA 6,124,198,8,103,24,8
1880 DATA 17,80,0,124,238,56,103
1890 DATA 25,197,229,6,80,221,126
1900 DATA 0,119,221,35,35,16,247
1910 DATA 225,193,16,218,201,0,144
1920 DATA 0,0,62,58,205,30,187
1930 DATA 192,62,60,205,30,187,194
1940 DATA 26,165,24,240,6,4,33
1950 DATA 79,165,17,0,169,205,140
1960 DATA 188,33,0,149,17,88,2
1970 DATA 1,0,0,62,2,205,152
1980 DATA 188,205,143,188,205,146,1
88
1990 DATA 201,6,4,33,79,165,17
2000 DATA 0,169,205,119,188,33,0
2010 DATA 144,205,131,188,205,122,1
88
2020 DATA 201,86,73,68,65,0,0
2030 DATA 38,38,46,25,205,117,187
2040 DATA 42,83,165,35,34,83,165
2050 DATA 55,17,16,39,35,62,47
2060 DATA 60,237,82,48,251,205,152
2070 DATA 165,17,232,3,60,237,82
2080 DATA 48,251,205,152,165,17,100
2090 DATA 0,60,237,82,48,251,205
2100 DATA 152,165,17,10,0,60,237
2110 DATA 82,48,251,205,152,165,133
2120 DATA 205,152,165,201,205,90,18
7
2130 DATA 62,47,32,1,35,25,35
2140 DATA 201,0,0,0,0,0,0
```




```
10 ;JUEGO DE LA VIDA
20          ORG  #A000
30          LD   HL,0
40          LD   (LATID),HL
50          LD   HL,#9000
60          LD   DE,#9001
70          LD   BC,1200
80          LD   (HL),0
90          LDIR
100         LD   HL,#90A5
110         LD   (POSCUR),HL
120 TEC:    LD   A,54
130         CALL #BB1E
140         RET  NZ
150         LD   A,36
160         CALL #BB1E
170         JR   Z,TECLO
180         CALL LOAD
190 TECLO:  XOR  A
200         CALL #BB1E
210         JR   Z,TECL1
220         CALL UP
230 TECL1:  LD   A,2
240         CALL #BB1E
250         JR   Z,TECL2
260         CALL DOWN
270 TECL2:  LD   A,8
280         CALL #BB1E
290         JR   Z,TECL3
300         CALL LEFT
310 TECL3:  LD   A,1
320         CALL #BB1E
330         JR   Z,TECL4
340         CALL RIGHT
350 TECL4:  LD   A,27
360         CALL #BB1E
370         JR   Z,TECL5
380         XOR  A
390         LD   (PINTI),A
400 TECL5:  LD   A,34
410         CALL #BB1E
420         JR   Z,TECL6
430         LD   A,1
440         LD   (PINTI),A
450 TECL6:  CALL PINBU
460         LD   A,9
470         CALL #BB1E
480         JP   NZ,INIC
490         JR   TEC
500 PINBU:  LD   HL,(POSCUR)
510         LD   A,(CONBY)
520         CP   0
530         JR   NZ,BYT1
540         SET  0,(HL)
550 BYT1:   CP   1
560         JR   NZ,BYT2
570         SET  1,(HL)
580 BYT2:   CP   2
590         JR   NZ,BYT3
600         SET  2,(HL)
610 BYT3:   CP   3
620         JR   NZ,BYT4
630         SET  3,(HL)
640 BYT4:   CP   4
650         JR   NZ,BYT5
660         SET  4,(HL)
670 BYT5:   CP   5
680         JR   NZ,BYT6
690         SET  5,(HL)
700 BYT6:   CP   6
710         JR   NZ,BYT7
720         SET  6,(HL)
730 BYT7:   CP   7
740         JR   NZ,BYT8
750         SET  7,(HL)
760 BYT8:   CALL PINTU
770         LD   A,(PINTI)
780         AND  A
790         JR   NZ,BORRA
800         RET
810 BORRA:  LD   HL,(POSCUR)
820         LD   A,(CONBY)
830         CP   0
840         JR   NZ,BITO
850         RES  0,(HL)
860 BITO:   CP   1
870         JR   NZ,BIT1
880         RES  1,(HL)
890 BIT1:   CP   2
900         JR   NZ,BIT2
910         RES  2,(HL)
920 BIT2:   CP   3
930         JR   NZ,BIT3
940         RES  3,(HL)
950 BIT3:   CP   4
960         JR   NZ,BIT4
970         RES  4,(HL)
980 BIT4:   CP   5
990         JR   NZ,BIT5
1000        RES  5,(HL)
1010 BIT5:  CP   6
1020        JR   NZ,BIT6
1030        RES  6,(HL)
1040 BIT6:  CP   7
1050        JR   NZ,BIT8
1060        RES  7,(HL)
1070 BIT8:  CALL PINTU
1080        RET
1090 UP:    LD   HL,(POSCUR)
1100        LD   DE,10
1110        SCF
1120        CCF
1130        SBC  HL,DE
1140        LD   (POSCUR),HL
1150        RET
1160 DOWN:  LD   HL,(POSCUR)
1170        LD   DE,10
1180        ADD  HL,DE
1190        LD   (POSCUR),HL
1200        RET
1210 LEFT:  LD   A,(CONBY)
1220        INC  A
1230        LD   (CONBY),A
1240        CP   8
1250        RET  NZ
1260 LEF1:  LD   HL,(POSCUR)
1270        DEC  HL
1280        LD   (POSCUR),HL
1290        XOR  A
1300        LD   (CONBY),A
1310        RET
1320 RIGHT: LD   A,(CONBY)
1330        DEC  A
1340        LD   (CONBY),A
1350        CP   255
1360        RET  NZ
1370        LD   HL,(POSCUR)
1380        INC  HL
1390        LD   (POSCUR),HL
1400        LD   A,7
1410        LD   (CONBY),A
1420        RET
1430 ;
1440 INIC:  LD   HL,#9500
1450 ;
1460        LD   DE,#9501
1470        LD   BC,600
1480        LD   (HL),0
1490        LDIR
1500        LD   HL,#9500
1510        LD   IX,#9000
1520        LD   BC,600
1530 MIRBU: LD   E,(IX)
1540        LD   D,0
1550        BIT  7,E
1560        JR   Z,PASBI1
1570        CALL MIR7
1580        CALL MIRBIT
1590        CALL Z,PON7
1600        JR   PASBI2
1610 PASBI1: CALL MIR7
1620        CALL MIRBI1
1630        CALL Z,PON7
1640 PASBI2: LD  D,0
1650        BIT  6,E
1660        JR   Z,PASBI3
1670        CALL MIR6
1680        CALL MIRBIT
1690        CALL Z,PON6
1700        JR   PASBI4
1710 PASBI3: CALL MIR6
1720        CALL MIRBI1
1730        CALL Z,PON6
1740 PASBI4: LD  D,0
```

```
1750          BIT   5,E
1760          JR    Z,PASBI5
1770          CALL  MIR5
1780          CALL  MIRBIT
1790          CALL  Z,PON5
1800          JR    PASBI6
1810 PASBI5:  CALL  MIR5
1820          CALL  MIRBI1
1830          CALL  Z,PON5
1840 PASBI6:  LD    D,0
1850          BIT   4,E
1860          JR    Z,PASBI7
1870          CALL  MIR4
1880          CALL  MIRBIT
1890          CALL  Z,PON4
1900          JR    PASBI8
1910 PASBI7:  CALL  MIR4
1920          CALL  MIRBI1
1930          CALL  Z,PON4
1940 PASBI8:  LD    D,0
1950          BIT   3,E
1960          JR    Z,PASBI9
1970          CALL  MIR3
1980          CALL  MIRBIT
1990          CALL  Z,PON3
2000          JR    PASB10
2010 PASBI9:  CALL  MIR3
2020          CALL  MIRBI1
2030          CALL  Z,PON3
2040 PASB10:  LD    D,0
2050          BIT   2,E
2060          JR    Z,PASB11
2070          CALL  MIR2
2080          CALL  MIRBIT
2090          CALL  Z,PON2
2100          JR    PASB12
2110 PASB11:  CALL  MIR2
2120          CALL  MIRBI1
2130          CALL  Z,PON2
2140 PASB12:  LD    D,0
2150          BIT   1,E
2160          JR    Z,PASB13
2170          CALL  MIR1
2180          CALL  MIRBIT
2190          CALL  Z,PON1
2200          JR    PASB14
2210 PASB13:  CALL  MIR1
2220          CALL  MIRBI1
2230          CALL  Z,PON1
2240 PASB14:  LD    D,0
2250          BIT   0,E
2260          JR    Z,PASB15
2270          CALL  MIR0
2280          CALL  MIRBIT
2290          CALL  Z,PON0
2300          JR    PASB16
2310 PASB15:  CALL  MIR0
2320          CALL  MIRBI1
2330          CALL  Z,PON0
2340 PASB16:  INC   IX
2350          INC   HL
2360          DEC   BC
2370          LD    A,B
2380          OR    C
2390          JP    NZ,MIRBU
2400          CALL  PINTA
2410          CALL  INCLAT
2420          LD    A,27
2430          CALL  #BB1E
2440          CALL  NZ,ESPER
2450          LD    A,54
2460          CALL  #BB1E
2470          JP    Z,INIC
2480          RET
2490 MIR7:    BIT   0,(IX-1)
2500          CALL  NZ,INCCON
2510          BIT   6,(IX+0)
2520          CALL  NZ,INCCON
2530          BIT   0,(IX+9)
2540          CALL  NZ,INCCON
2550          BIT   7,(IX+10)
2560          CALL  NZ,INCCON
2570          BIT   6,(IX+10)
2580          CALL  NZ,INCCON
2590          BIT   6,(IX-10)
2600          CALL  NZ,INCCON
2610          BIT   7,(IX-10)
2620          CALL  NZ,INCCON
2630          BIT   0,(IX-11)
2640          CALL  NZ,INCCON
2650          RET
2660 MIR6:    BIT   7,(IX+0)
2670          CALL  NZ,INCCON
2680          BIT   5,(IX+0)
2690          CALL  NZ,INCCON
2700          BIT   7,(IX-10)
2710          CALL  NZ,INCCON
2720          BIT   6,(IX-10)
2730          CALL  NZ,INCCON
2740          BIT   5,(IX-10)
2750          CALL  NZ,INCCON
2760          BIT   7,(IX+10)
2770          CALL  NZ,INCCON
2780          BIT   6,(IX+10)
2790          CALL  NZ,INCCON
2800          BIT   5,(IX+10)
2810          CALL  NZ,INCCON
2820          RET
2830 MIR5:    BIT   6,(IX+0)
2840          CALL  NZ,INCCON
2850          BIT   4,(IX+0)
2860          CALL  NZ,INCCON
2870          BIT   6,(IX-10)
2880          CALL  NZ,INCCON
2890          BIT   5,(IX-10)
2900          CALL  NZ,INCCON
2910          BIT   4,(IX-10)
2920          CALL  NZ,INCCON
2930          BIT   6,(IX+10)
```



```
2940          CALL  NZ,INCCON
2950          BIT   5,(IX+10)
2960          CALL  NZ,INCCON
2970          BIT   4,(IX+10)
2980          CALL  NZ,INCCON
2990          RET
3000 MIR4:    BIT   5,(IX+0)
3010          CALL  NZ,INCCON
3020          BIT   3,(IX+0)
3030          CALL  NZ,INCCON
3040          BIT   5,(IX-10)
3050          CALL  NZ,INCCON
3060          BIT   4,(IX-10)
3070          CALL  NZ,INCCON
3080          BIT   3,(IX-10)
3090          CALL  NZ,INCCON
3100          BIT   5,(IX+10)
3110          CALL  NZ,INCCON
3120          BIT   4,(IX+10)
3130          CALL  NZ,INCCON
3140          BIT   3,(IX+10)
3150          CALL  NZ,INCCON
3160          RET
3170 MIR3:    BIT   4,(IX+0)
3180          CALL  NZ,INCCON
3190          BIT   2,(IX+0)
3200          CALL  NZ,INCCON
3210          BIT   4,(IX-10)
3220          CALL  NZ,INCCON
3230          BIT   3,(IX-10)
3240          CALL  NZ,INCCON
3250          BIT   2,(IX-10)
3260          CALL  NZ,INCCON
3270          BIT   4,(IX+10)
3280          CALL  NZ,INCCON
3290          BIT   3,(IX+10)
3300          CALL  NZ,INCCON
3310          BIT   2,(IX+10)
3320          CALL  NZ,INCCON
3330          RET
3340 MIR2:    BIT   3,(IX+0)
3350          CALL  NZ,INCCON
3360          BIT   1,(IX+0)
3370          CALL  NZ,INCCON
3380          BIT   3,(IX-10)
3390          CALL  NZ,INCCON
3400          BIT   2,(IX-10)
3410          CALL  NZ,INCCON
3420          BIT   1,(IX-10)
3430          CALL  NZ,INCCON
3440          BIT   3,(IX+10)
3450          CALL  NZ,INCCON
3460          BIT   2,(IX+10)
3470          CALL  NZ,INCCON
3480          BIT   1,(IX+10)
3490          CALL  NZ,INCCON
3500          RET
3510 MIR1:    BIT   2,(IX+0)
3520          CALL  NZ,INCCON
3530          BIT   0,(IX+0)
3540          CALL  NZ,INCCON
3550          BIT   2,(IX-10)
3560          CALL  NZ,INCCON
3570          BIT   1,(IX-10)
3580          CALL  NZ,INCCON
3590          BIT   0,(IX-10)
3600          CALL  NZ,INCCON
3610          BIT   2,(IX+10)
3620          CALL  NZ,INCCON
3630          BIT   1,(IX+10)
3640          CALL  NZ,INCCON
3650          BIT   0,(IX+10)
3660          CALL  NZ,INCCON
3670          RET
3680 MIR0:    BIT   1,(IX+0)
3690          CALL  NZ,INCCON
3700          BIT   7,(IX+1)
3710          CALL  NZ,INCCON
3720          BIT   1,(IX-10)
3730          CALL  NZ,INCCON
3740          BIT   0,(IX-10)
3750          CALL  NZ,INCCON
3760          BIT   7,(IX-9)
3770          CALL  NZ,INCCON
3780          BIT   1,(IX+10)
3790          CALL  NZ,INCCON
3800          BIT   0,(IX+10)
3810          CALL  NZ,INCCON
3820          BIT   7,(IX+11)
3830          CALL  NZ,INCCON
3840          RET
3850          RET
```

```
3860 INCCON: INC  D
3870         RET
3880 MIRBI1: LD   A,D
3890         CP   3
3900         RET  Z
3910         RET
3920 MIRBIT: LD   A,D
3930         CP   2
3940         RET  Z
3950         CP   3
3960         RET  Z
3970         RET
3980 PON7:   SET  7,(HL)
3990         RET
4000 PON6:   SET  6,(HL)
4010         RET
4020 PON5:   SET  5,(HL)
4030         RET
4040 PON4:   SET  4,(HL)
4050         RET
4060 PON3:   SET  3,(HL)
4070         RET
4080 PON2:   SET  2,(HL)
4090         RET
4100 PON1:   SET  1,(HL)
4110         RET
4120 PONO:   SET  0,(HL)
4130         RET
4140 PINTA:  LD   HL,#9500
4150         LD   DE,#9000
4160         LD   BC,600
4170         LDIR
4180 PINTU:  LD   HL,#4000
4190         LD   DE,#4001
4200         LD   BC,#4000
4210         LD   (HL),0
4220         LDIR
4230         CALL MIRA
4240         CALL IMPRE
4250         RET
4260 MIRA:   LD   DE,#4000
4270         LD   HL,#9000
4280         LD   B,60
4290 LLL2:   PUSH BC
4300         LD   (POSIN),DE
4310         LD   B,10
4320 LLL1:   XOR  A
4330         BIT  7,(HL)
4340         CALL NZ,PON67
4350         LD   (DE),A
4360         XOR  A
4370         INC  DE
4380         BIT  6,(HL)
4390         CALL NZ,PON67
4400         LD   (DE),A
4410         XOR  A
4420         INC  DE
4430         BIT  5,(HL)
4440         CALL NZ,PON67
4450         LD   (DE),A
4460         XOR  A
4470         INC  DE
4480         BIT  4,(HL)
4490         CALL NZ,PON67
4500         LD   (DE),A
4510         XOR  A
4520         INC  DE
4530         BIT  3,(HL)
4540         CALL NZ,PON67
4550         LD   (DE),A
4560         XOR  A
4570         INC  DE
4580         BIT  2,(HL)
4590         CALL NZ,PON67
4600         LD   (DE),A
4610         XOR  A
4620         INC  DE
4630         BIT  1,(HL)
4640         CALL NZ,PON67
4650         LD   (DE),A
4660         XOR  A
4670         INC  DE
4680         BIT  0,(HL)
4690         CALL NZ,PON67
4700         LD   (DE),A
4710         INC  DE
4720         INC  HL
4730         DJNZ LLL1
4740         PUSH HL
4750         LD   HL,(POSIN)
4760         LD   BC,80
4770         LDIR
4780         LD   HL,80
4790         ADD  HL,DE
4800         EX   DE,HL
4810         POP  HL
4820         POP  BC
4830         DJNZ LLL2
4840         LD   (POSIN),DE
4850         RET
4860 ;
4870 POSIN:  DEFS 2
4880 ;
4890 PON67:  SET  6,A
4900         SET  7,A
4910         SET  5,A
4920         SET  4,A
4930         SET  3,A
4940         RET
4950 ;RUTINA-IMPRESION
4960 ;H-POSICION-VERTICAL-INICIO-1
4970 ;L-POSICION-HORIZONTAL-INICIO-1
4980 ;DE-DIRECCION-GRAFICO
4990 ;
5000 IMPRE:  LD   HL,#0101
5010         LD   DE,#4000
5020         PUSH DE
5030         EX   DE,HL
5040         LD   HL,#C000-80
5050         LD   B,D
5060         LD   D,0
5070         DEC  E
5080         ADD  HL,DE
5090         LD   DE,80
5100 S_BUC:  ADD  HL,DE
5110         DJNZ S_BUC
5120         POP  IX
5130         LD   B,200
5140         JR   COLOC
5150 P_BUC:  LD   A,H
5160         AND  56
5170         CP   56
5180         JR   Z,P_PAS
5190         LD   A,H
5200         ADD  A,8
5210         LD   H,A
5220         JR   COLOC
5230 P_PAS:  LD   DE,0080
5240         LD   A,H
5250         XOR  56
5260         LD   H,A
5270         ADD  HL,DE
5280 COLOC:  PUSH BC
5290         PUSH HL
5300         LD   B,80
5310 P_BUC1: LD   A,(IX+0)
5320         LD   (HL),A
5330         INC  IX
5340         INC  HL
5350         DJNZ P_BUC1
5360         POP  HL
5370         POP  BC
5380         DJNZ P_BUC
5390         RET
5400         DEFS 0
5410 POSCUR: DEFW #9000
5420 CONBY:  DEFB 0
5430 PINTI:  DEFB 0
5440 ESPER:  LD   A,58
5450         CALL #BB1E
5460         RET  NZ
5470         LD   A,60
5480         CALL #BB1E
5490         JP   NZ,SAVE
5500         JR   ESPER
5510 SAVE:   LD   B,4
5520         LD   HL,NAME
5530         LD   DE,#A900
5540         CALL #BC8C
5550         LD   HL,#9500
5560         LD   DE,600
5570         LD   BC,0
5580         LD   A,2
5590         CALL #BC98
5600         CALL #BC8F
5610         CALL #BC92
5620         RET
5630 LOAD:   LD   B,4
5640         LD   HL,NAME
5650         LD   DE,#A900
5660         CALL #BC77
5670         LD   HL,#9000
5680         CALL #BC83
5690         CALL #BC7A
5700         RET
5710 NAME:   DEFM "VIDA"
5720 LATID:  DEFW 0
5730 INCLAT: LD   H,38
5740         LD   L,25
5750         CALL #BB75
5760         LD   HL,(LATID)
5770         INC  HL
5780         LD   (LATID),HL
5790         SCF
5800         LD   DE,10000
5810         INC  HL
5820         LD   A,47
5830 DMIL:   INC  A
5840         SBC  HL,DE
5850         JR   NC,DMIL
5860         CALL PRINT
5870         LD   DE,1000
5880 MIL:    INC  A
5890         SBC  HL,DE
5900         JR   NC,MIL
5910         CALL PRINT
5920         LD   DE,100
5930 CIEN:   INC  A
5940         SBC  HL,DE
5950         JR   NC,CIEN
5960         CALL PRINT
5970         LD   DE,10
5980 DIEZ:   INC  A
5990         SBC  HL,DE
6000         JR   NC,DIEZ
6010         CALL PRINT
6020         ADD  A,L
6030         CALL PRINT
6040         RET
6050 PRINT:  CALL #BB5A
6060         LD   A,47
6070         JR   NZ,PAS
6080         INC  HL
6090 PAS:    ADD  HL,DE
6100         INC  HL
6110         RET
```

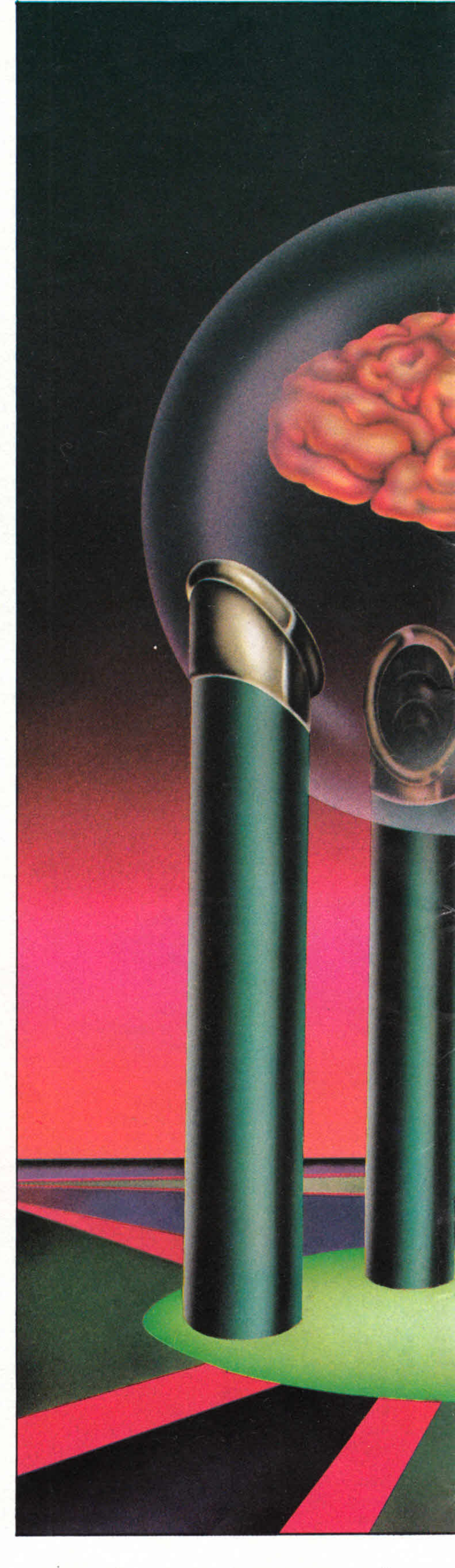
ETIQUETAS

BIT1	A0C6	BIT2	A0CC		
BIT4	A0D8	BIT5	A0DE		
BIT8	A0EA	BORRA	A0B4		
BYT2	A086	BYT3	A08C		
BYT5	A098	BYT6	A09E		
BYT8	A0AA	CIEN	A580		
CONBY	A508	DIEZ	A58B		
DOWN	A0FC	ESPER	A50A		
INCCON	A407	INCLAT	A555		
LATID	A553	LEF1	A111		
LLL1	A45C	LLL2	A455		
MIL	A575	MIRO	A3CD		
MIR2	A35B	MIR3	A322		
MIR5	A2B0	MIR6	A277		
MIRA	A44D	MIRBI1	A409		
MIRBU	A14B	NAME	A54F		
PASB10	A1D0	PASB11	A1E1		
PASB13	A1FB	PASB14	A204		
PASB16	A21E	PASBI1	A15F		
PASBI3	A179	PASBI4	A182		
PASBI6	A19C	PASBI7	A1AD		
PASBI9	A1C7	PINBU	A074		
PINTA	A42E	PINTI	A509	PINTU	A439
PONO	A42B	PON1	A428	PON2	A425
PON3	A422	PON4	A41F	PON5	A41C
PON6	A419	PON67	A4B8	PON7	A416
POSCUR	A506	POSIN	A4B6	PRINT	A598
P_BUC	A4DF	P_BUC1	A4F8	P_PAS	A4EC
RIGHT	A11D	SAVE	A51A	S_BUC	A4D6
TEC	A019	TECLO	A029	TECL1	A032
TECL2	A03C	TECL3	A046	TECL4	A050
TECL5	A05B	TECL6	A067	UP	A0EE

FORTH: POTENCIA Y VELOCIDAD EN ALTO NIVEL

David Sopuerta

Como todos sabemos, el lenguaje utilizado por excelencia para programar nuestros ordenadores caseros ha sido el Basic. Pero los aficionados inquietos, se habrán dado cuenta rápidamente que no es el lenguaje de programación ideal para que nuestro Amstrad actúe de una manera eficaz en todas las ocasiones y, que algunas aplicaciones como, por ejemplo, los juegos de arcade y similares, están pidiendo a gritos que utilicemos un sistema más rápido y compacto a la hora de codificarlos.



La primera solución que se nos ocurre es escribir este tipo de programas en código máquina. ¡Qué problema! A la mayoría de nosotros puede que hacerlo nos resulte todavía bastante difícil y penoso ya que se trata de un lenguaje en el que hay que **«hilar»** muy fino, teniendo muy claro qué es lo que queremos hacer y conociendo con precisión las instrucciones internas del micro.

Además, los programas generados son largos y de difícil seguimiento en caso de error. Tiene un gran número de instrucciones elementales entre las que nos resultará relativamente fácil perdernos. Así que de momento...

Un camino más fácil y entretenido es utilizar alguno de los muchos lenguajes de alto nivel que ya están a disposición de nuevo **Amstrad.** Entre ellos están incluidos el Pascal, el Forth y el Logo y, si su ordenador está equipado con una unidad de disco, también podrá hacer sus programas en Lisp, Prolog, Fortran, C y muchos más.

Cada uno de ellos tiene sus ventajas e inconvenientes en las diferentes aplicaciones que les demos. Mientras un lenguaje puede parecer ideal para una cosa en concreto, a lo mejor resulta ser demasiado lento o necesitar demasiada memoria en otras ocasiones.

En pocas palabras. Lo que realmente estamos necesitando es un lenguaje «ideal» que, en conjunto, sea suficientemente rápido para cualquier requisito, que no «despilfarre» demasiada memoria y que, por supuesto, sea relativamente fácil de aprender y utilizar.

Forth: historia de la eficacia

El lenguaje que, en principio, se adapta mejor a estos requisitos es el Forth y, no debe sorprenderle que sea el segundo en popularidad entre los usuarios de ordenadores caseros. Es sencillo, compacto, pensado para aplicaciones de uso general, ideal para emplearlo en multitud de problemas y su aprendizaje no entraña ninguna dificultad, a pesar de su estructura y vocabulario poco usuales.

El Forth nació alrededor de 1969 y en un principio se utilizó para controlar los complejos movimientos de los grandes telescopios. Desde entonces ha tenido una amplia y variada gama de usuarios para una no menos extensa galería de aplicaciones.

Posee alguna de las características avanzadas de los lenguajes de alto nivel, tales como estructuras de bucles y análisis de condiciones muy complejas, y genera programas muy uniformes que se ejecutan a una gran velocidad, aproximadamente 10 veces más deprisa que el Basic. A esto hay que añadir que podemos modificar y ampliar el lenguaje dotándolo de nuevas palabras «clave» a medida que vayan surgiendo a lo largo de cualquier aplicación.



Nuestra primera tarea, antes de comenzar a desentrañar las maravillas de este nuevo lenguaje, es, por supuesto, teclear el programa que nos servirá para ejecutar en el **Amstrad** las instrucciones que escribamos en Forth.

Un intérprete para aprender

Dese cuenta de algo importante. Este programa no nos genera realmente una **versión del Forth**, simplemente simula esta operación. Su misión es convertir cada nueva palabra en un código que responda a un formato interno especial que se corresponde con unas instrucciones Basic que son las que realmente realizan el trabajo.

Y como a continuación el **Amstrad** ha de interpretar estas sentencias Basic, no se extrañe que en esta ocasión cualquier programa creado en Forth se ejecutará bastante lentamente. Pero no se desespere. Por lo demás esta versión nos permite utilizar un Forth bastante parecido al que se usa realmente. A pesar de su lentitud, nos permitirá hacer experimentos con este potente lenguaje empleando las técnicas sugeridas en este artículo. Cuando su aprendizaje haya terminado ya estaremos capacitados para decidir si Forth es **«nuestro»** lenguaje. De ser así, ya sabe lo que le tocará hacer: adquirir una de las versiones comerciales disponibles para el **Amstrad.**

Al principio puede resultarnos un poco incómodo de utilizar debido principalmente a que emplea una notación diferente a la que nosotros estamos acostumbrados. En vez de escribir instrucciones en la forma que lo hace el Basic, todas las sentencias Forth, o **«palabras»** como de ahora en adelante las conoceremos, necesitan tener sus argumentos —o números con los que trabajan— **«delante»** de cada comando y no después como es el caso de la mayor parte de los lenguajes.

El Forth: es un lenguaje basado en el concepto de stack

Por ejemplo, si queremos sumar dos números y después imprimir su resultado, lo conseguiremos con la siguiente instrucción:

```
PRINT 3+8
```

Pero en Forth no se haría así. Tendremos que escribir algo parecido a:

```
3 8+.
```

donde el punto (.) es la «palabra» Forth empleada para visualizar en la pantalla. Esta forma de escribir las instrucciones es lo que se conoce como «Notación Polaca Inversa» y le aseguramos que no es tan enrevesada como parece, así que no se sienta desanimado por esta novedad.

Sin embargo, no es la única diferencia existente. El Forth trabaja empleando una parte de la memoria llamada «stack» o pila para guardar los valores que en un determinado momento van a tener los argumentos que vayamos a pasar a una **«palabra».**

Su funcionamiento es muy sencillo. Imagine que está recogiendo platos y **«apilándolos»** uno encima del otro. Es evidente que el último plato que hayamos tomado será el que coloquemos en la parte superior de la **«pila».** Una vez colocados, necesitamos coger uno. *¿Cuál será?*

A menos que sea un malabarista e intente demostrarlo sacando uno de los platos de abajo, lo más normal y lógico es que el elegido sea precisamente el que está colocado encima de todos. Así sucede con el «stack», el último dato que hayamos colocado en él es generalmente el primero que después vamos a sacar de allí.

Es lo que los técnicos han bautizado como memoria **«LIFO»** (o **«last input-first output»**) que quiere decir: el último elemento que entra es el primero que sale.

La manera en la que el Forth interpreta las instrucciones, o **«palabras»,** que escribimos anteriormente es bastante sencilla. Coloca en el stack los números que vamos a sumar en el orden que se los hemos dado —primero el 3 y encima el 8. A continuación se ejecuta la palabra Forth +.

El modo de operación es muy semejante en la mayor parte de las sentencias Forth: se sacan los dos números del stack, se procesan en la forma oportuna y el resultado se vuelve a guardar en el stack para que pueda ser manejado por sucesivas palabras.

En este caso, se cogen los dos datos que ocupan las posiciones más altas de la pila, se suman y se devuelve este dato al stack.

A continuación la siguiente palabra (.) extrae el número que ahora esté colocado encima (el resultado de la operación anterior) y lo saca en la pantalla. El Forth además imprime el mensaje de **«OK»** para convencernos de que la instrucción se ha ejecutado sin errores. Observe y tenga siempre en cuenta que tanto los datos como las palabras Forth han de estar separadas por un espacio en blanco. ¡No lo olvide!

Estos comandos dejan el stack exactamente igual que estaba antes de ejecutarlos. Por nuestra parte le damos un consejo: es siempre conveniente que así ocurra ya que de esta forma permitimos a las **«palabras»** siguientes, operar con los valores que se colocaron en la pila antes de ejecutarse la anterior secuencia de instrucciones.

El límite superior del stack siempre contiene el último número introducido y si metemos en él un nuevo valor, el anterior queda debajo de modo que el elemento que hemos almacenado más recientemente es el que ocupará la parte más alta de la pila, de ahí que se le llame **«top of stack»** (superior de la pila).

Quizá nos llame la atención, por lo que hemos visto hasta ahora, que el Forth sólo sea capaz de sumar dos números y no tres, cuatro o todos los que queramos. Pero es lógico, la «palabra« Forth «+» opera siempre solamente con dos números (al fin y al cabo es lo que nosotros hacemos con cada una de sus ci-

fras) y a continuación devuelve al stack un único resultado. Y esta forma de trabajo no sólo es válida para la suma sino que podemos extenderlo al resto de las operaciones aritméticas empleadas en Forth, tales como multiplicaciones y divisiones.

Pero este modo de desarrollar el lenguaje no nos impide utilizar expresiones más complejas. Solamente será necesario poner un poco más cuidado en lo que estamos haciendo a la hora de decidir cómo organizarlas según esta nueva notación.

Evaluación de expresiones en Forth

Pongamos un ejemplo. Si queremos evaluar en Forth la expresión:

15+2×9

debemos comenzar multiplicando 2 por 9 (recuerde la prioridad de operaciones) para encontrar un resultado intermedio que sumado luego a 15, nos dé el valor final de la expresión.

Otra forma de calcularlo sería sumando primero 15 y 2 y después multiplicar el resultado por 9. Observe que la solución obtenida es completamente diferente de la anterior. ¿Cuál es la buena?

Recuerde que en Basic existe un orden de ejecución en las operaciones matemáticas y que el producto es más prioritario que la suma. Por tanto, el Basic y la mayoría de los lenguajes utilizarán el primer método de evaluación.

Sigámosle también en Forth. Para multiplicar 2 y 9 tendremos que teclear:

2 9 *

y nos dejará el resultado (18) en el elemento superior del stack. Podemos comprobarlo visualizándole en la pantalla usando la palabra **«.»** pero tenga en cuenta que, si así lo hacemos desaparecerá de la pila. De manera que como vamos a necesitar este valor en la siguiente parte del cálculo es mejor que lo dejemos donde estaba.

Después necesitamos sumar 15 al elemento superior de la pila. Pues teclee:

15+

y colocará el resultado en el nuevo «TOS» (top of stack) pudiéndolo imprimir a continuación. La primitiva expresión Basic se ha convertido en:

2 9*15+.

que es el comando Forth que nos va a sacar en la pantalla el resultado correcto de la evaluación: 33.

Sobre los cálculos matemáticos queremos puntualizar algo. La mayor parte de las versiones del Forth tiene unos operadores aritméticos que utilizan solamente nùmeros enteros y además dan resultados que son también enteros.

Escriba ahora:

9 4/.

que tiene toda la pinta de ser el equivalente en Forth a:

PRINT 9/4

y la respuesta obtenida es 2 en lugar de 2.25 que sería el valor exacto. De esta forma confirmamos que el número que nos devuelve la operación es también entero. Así que de momento queda bastante claro que no podemos usar números con punto decimal en esta versión del lenguaje.

Pero también está restringido el rango de los números. Intente teclear:

1000 200 *.

Parece que la respuesta que aparece en la pantalla no está muy de acuerdo con lo que esperábamos. La aritmética en Forth está pensada para ser utilizada en micros, del mismo modo que el lenguaje. Por tanto, trabajará con números que puedan ser representados con 16 bits *(o dígitos binarios)* o lo que es lo mismo, con valores comprendidos dentro del rango de —32768 a 32767. De ahí el mensaje, aunque sea ficticio, de **«stack lleno»:** la pila no está llena, lo que se produce es un **«overflow»** o desbordamiento de la capacidad numérica de su micro.

Esta notación aritmética no es tan compleja como parece. El mejor camino para utilizarla con soltura es intentar emplearla muchas veces para hacer en Forth todas las operaciones que se nos ocurran. Después de realizar unas cuantas prácticas verá que su uso es casi tan sencillo como la empleada en la aritmética habitual con una ventaja: es mucho más potente.

La potencia del Forth está en crear nuevas órdenes en Forth

Sin embargo, donde radica el poder real del lenguaje Forth es en el hecho de poder definir nuevas «palabras» que se añaden inmediatamente a su vocabulario básico así como definir con otro nombre las ya existentes.

La **«palabra»** Forth es algo equivalente a las subrutinas, o partes de un programa Basic a las que se salta tras un GOSUB y siempre terminan con RETURN.

De un modo sencillo, podemos decir que una **«palabra»** es una serie de instrucciones, y sus correspondientes parámetros, que están agrupadas bajo un nombre que incorporamos, tras definirla, al vocabulario Forth.

Para ejecutar esta secuencia basta con invocarla, o llamarla, por su nombre y el programa salta a la serie de instrucciones, así bautizadas, y las ejecuta.

Supongamos que preferimos utilizar palabras castellanas como operadores aritméticos en lugar de los signos matemáticos y también emplear ESCRIBIR en lugar del punto Forth. Todo lo que hay que hacer es definir las nuevas palabras.

Para ello escribimos:

: SUMAR+;
:RESTAR—;
:MULTIPLICAR*;

para las próximas palabras aritméticas, y

: ESCRIBIR.;

para conseguir que nos aparezcan los resultados en la pantalla.

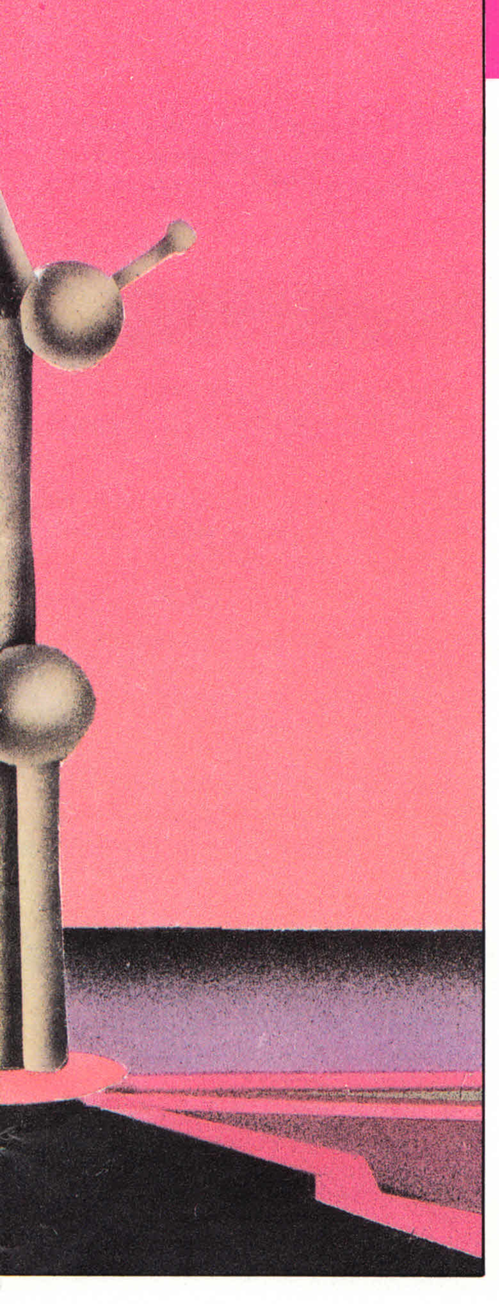
Con estas definiciones, la expresión aritmética que anteriormente analizamos podría transformarse en:

2 9 MULTIPLICAR 15 SUMAR ESCRIBIR

obteniendo el mismo resultado en ambos casos.

La primitiva expresión sigue siendo válida aunque actualmente hayamos definido otras, con nombre diferente, pero que realizan las mismas operaciones.

Todas las nuevas palabras Forth que que-



ramos crear se definen de la misma forma. Comenzamos poniendo dos puntos seguidos por el nombre que queremos darle, a continuación ponemos la serie de instrucciones que queremos agrupar y cerramos la definición con un punto y coma.

Los dos puntos indican al Forth que vamos a definir una nueva palabra y deben ir seguidos forzosamente por el nombre que queramos darle.

Después viene la secuencia de parámetros y palabras Forth ya definidas que se ejecutarán al llamar a la nueva. No es necesario que sean palabras standard del lenguaje —tales como los signos+ y —del ejemplo anterior— sino que podemos usar también cualquier palabra de las que nosotros ya hemos creado.

Finalmente, la definición se cierra con un punto y coma. De esta manera podemos conseguir construir un lenguaje completo y compacto, en el que se emplean palabras standard del lenguaje para crear otras nuevas, y después utilizar las nuevas para otras sucesivas definiciones. El lenguaje crece de esta manera indefinidamente.

Los programas siguen el método de diseño «Top-down»

Así es exactamente como se genera un programa en Forth. Dividimos un problema general en grupos de acciones más particulares. Estos últimos ya se pueden definir utilizando las palabras standard Forth.

Y es en este punto donde comenzamos para luego ir definiendo otras cada vez más complejas. Al final llegaremos a un programa completo compuesto por una sola palabra que, al teclearla, hace que se ejecuten todas las asociadas.

Si seguimos usando el ejemplo anterior, podemos definir la siguiente palabra:

: CALCULO 2 9 MULTIPLICAR 15 SUMAR ESCRIBIR;

y ahora, con sólo teclear:

CALCULO

se ejecutarán todas las palabras agrupadas bajo este nombre, así como también las standard asociadas a ellas.

Por supuesto que los programas Forth no consisten únicamente en la evaluación de expresiones aritméticas —como en los casos anteriores— sino que también tienen a su disposición otras palabras que nos permitirán realizar cosas bastante más interesantes y atractivas.

Todas estas palabras básicas, incluidas en cada una de las versiones Forth que existen, es lo que se conoce como **«vocabulario»** del lenguaje, y en él están recogidas todas las que nos van a permitir utilizar variables, implementar bucles, explorar el teclado y un montón de cosas que son necesarias para que sus programas sean más complejos y le muestren la auténtica potencia del lenguaje.

Algunas órdenes del intérprete

Si quiere, puede hacer que le aparezca en la pantalla todo el vocabulario básico completo empleando el comando:

*VLIST

Y no sólo eso. También podemos visualizar todas las palabras que hayamos definido nosotros mismo. Si no lo ha hecho, le sugerimos que teclee las nuevas palabras de creación propia, para que por lo menos exista alguna.

A continuación teclee:

*LIST

y podrá conocer todas las que tiene a su disposición.

Pero empleando este comando puede también ver la serie de instrucciones que forman parte de cualquiera de ellas. Bastará con escribirlo seguido del nombre que queremos conocer y su listado estará **«servido»** .

¿Qué ocurre si por equivocación definimos dos veces una misma palabra? Sencillamente esta versión del Forth no acepta la segunda y nos da un mensaje de error:

Palabra ya definida

En otras, sí lo admite sin problemas. Guarda la segunda palabra además de la anterior. Pero cuando invocamos el nombre con el que está definida, el Forth accede a la más reciente de todas las que hayamos hecho sin tener en cuenta ninguna de las más antiguas.

Con este intérprete no se nos dará nunca este caso, ya que, como hemos dicho, no admitirá dos definiciones bajo el mismo nombre.

Entonces si queremos corregir o modificar una de las ya existentes el camino más cómodo a seguir sería listar la palabra para tener constancia de ella en la pantalla (no es necesario aunque sí muy conveniente). Después borrar del diccionario la que queramos cambiar mediante el comando:

*FORGET palabra

que le dirá al Forth que elimine de su diccionario la definición que hayamos hecho de «palabra».

A continuación creamos una nueva, auxiliándonos del listado existente en la pantalla con el cursor de copia, incluyendo las modificaciones que queramos hacer.

Sin embargo, para utilizar *FORGET debemos pensar muy bien lo que estamos haciendo ya que no sólo nos elimina la palabra que le hayamos dicho, sino que también borra todas las que estén definidas con posterioridad al nombre que sigue a *FORGET. Así que, ¡ojo con lo que borramos!

Hasta este momento debe tener por lo menos una pequeña idea de qué es lo que hay que hacer para trabajar con el Forth, pero seguro que todavía no tiene una noción clara de cómo podemos hacer un programa: no conocemos lo que hace cada una de las instrucciones del vocabulario standard del lenguaje.

Podemos dividirlas en grupos. El primero es el compuesto por todas las instrucciones encargadas de mantener en orden el stack cuando éste contiene parámetros. Son las instrucciones de tratamiento o manejo de la pila.

Instrucciones de manejo de la pila

Suponga que tenemos dos valores colocados en el stack y queremos hacer una división entre ellos. Por ejemplo:

6 2/.

nos calcularía el cociente entre 6 y 2 y nos lo sacaría en la pantalla.

Pero quizás el resultado que a nosotros nos interesaba es el inverso, o sea, el obtenido al

dividir 2 entre 6. La forma inmediata de hacerlo sería escribir:

2 6/.

No siempre es posible hacer esto, ya que en la mayoría de las ocasiones no conocemos los valores contenidos en el stack. Ahora bien, el Forth dispone de un operador que intercambia los dos valores superiores de la pila. En nuestro caso ya podríamos realizar el inverso de la operación anterior mediante:

6 2 SWAP/.

Para comprobar que con esta instrucción existe intercambio sin variar para nada el resto de los elementos del stack, teclee:

1 2 3 SWAP...

y aparecerán en la pantalla estos números pero con el 2 y el 3 cambiados de orden, o sea:

2 3 1

Pero no es este el único operador que maneja los elementos de la pila. ROT hace que giren los tres superiores. Coloca el tercero en la posición más alta y desplaza a los otros dos un lugar más abajo. Como siempre, la manera más efectiva de ver cómo funciona es un ejemplo práctico, así que escriba:

1 2 3 ROT...

El orden como quedan colocados es el que aparece en la pantalla y vemos que el 1 es el que está colocado encima de los otros dos. *¿Comprendido?*

Otro operador utilizado en Forth es DUP. Su misión es simplemente duplicar la última entrada que haya existido en el stack. Si quiere calcular el cuadrado de un número sería suficiente con hacer:

12 DUP *.

para que nos aparezca 144 (el cuadrado de 12) en la pantalla.

Si en lugar de esto lo que necesitamos es duplicar el elemento que está inmediatamente debajo del superior, el Forth posee un operador que realiza este trabajo: OVER. Compruébelo tecleando:

1 2 OVER...

que copiaría el 1 encima de 2 (elemento superior en ese preciso instante) e imprimiría después los tres elementos existentes sacándolos de la pila y dejándola vacía.

El último de los operadores de manejo de stack es DROP, que saca del mismo la última entrada de datos. Ejemplo al canto:

1 2 DROP...

Y éstos son los más importantes. Su conocimiento y manejo pueden resultarnos un poco liosos al principio pero poco a poco nos iremos habituando a ello y podremos **«saborear»** sus ventajas.

Las estructuras del control del lenguaje son muy potentes

Pero el Forth no sólo dispone de operadores aritméticos o de manejo del stack, sino que además posee unas estructuras de control clásicas en otros lenguajes que le dan una gran potencia y versatilidad. Veámoslas.

La sentencia que nos permitirá alterar el orden de ejecución de comandos dependiendo de que se cumpla o no una determinada condición es IF... THEN... ELSE. Su forma general es:

< condición > IF < acción-1 >
ELSE < acción-2 >
THEN < continuar >

Su modo de funcionamiento es como sigue. La evaluación de la condición deja un «falso» o «verdadero» en el stack. Este resultado se analiza y si es verdadero se ejecuta la acción-1, si no, realiza la acción-2 y después continua el programa en el punto que sigue a THEN.

El Programa I sería un ejemplo de utilización de esta estructura. Para utilizarlo bastaría con teclear la nota seguida de la palabra **«NOTA»** una vez que haya sido definida para que se nos informe de si hay aprobado o no.

Programa I

```
: NOTA 5 < IF. "SUSPENSO" ELSE." PROBADO" THEN CR." CORECTO ?";
```

El Forth dispone también de una serie de estructuras de bucle que hace que se repitan una serie de instrucciones durante un determinado número de veces. Es el conocido FOR... NEXT del lenguaje Basic.

< valor final > < valor inicial > DO < acción > LOOP

Con él estamos ejecutando acción desde que el **«índice»** es igual al valor inicial hasta que alcanza el valor final.

La palabra clave LOOP hace que volvamos a DO mientras no se haya alcanzado el valor final. Le sugerimos que intente hacer variaciones sobre este pequeño ejemplo, Programa II, para clarificar sus ideas.

Programa II

```
: BUCLE1 12 0 DO. «ESTO ES UN BUCLE QUE SE REPITE 12 VECES» CR LOOP;
```

Podemos acceder al índice del bucle mediante la palabra Forth **«I»** que obtiene el valor en curso de la variable índice del bucle DO y le sitúa en la parte superior del stack. Si teclea el Programa III lo verá en funcionamiento.

Programa III

```
: BUCLE 13 1 DO I. SPACE. «ESTO ES UN BUCLE QUE SE REPITE 12 VECES» CR LOOP;
```

Si deseamos que el índice no se incremente de 1 en 1 sino con un valor diferente, se puede utilizar la estructura:

DO... número + LOOP

poniendo en **«número»** el incremento deseado, tanto positivo como negativo. Pero ésta es una facilidad que no contempla nuestro intérprete Forth. ¡Es una pena!

Existen también otra clase de bucles que van a repetir una serie de comandos dependiendo de que se cumpla o no una condición. En Forth este tipo de bucle puede ser de dos formas.

He aquí la primera de ellas:

El bucle WHILE nos va a permitir repetir una serie de acciones mientras se esté cumpliendo una determinada condición. Su forma general es:

BEGIN < condición > WHILE < acción >
REPEAT

La palabra BEGIN indica simplemente el punto donde comienza el bucle. Después la condición deja un valor en el stack dependiendo que se cumpla o no.

WHILE analiza dicho valor y si no es cero *(falso)* ejecuta la **«acción».** REPEAT nos devuelve otra vez a BEGIN.

Si la condición no se cumple y, por tanto, el valor que coloca en el stack es cero, entonces no se ejecuta «acción» y se continúa por los siguiente a la palabra REPEAT.

El Programa IV es un ejemplo de la utilización de este tipo de estructuras.

Programa IV

```
: BUCLES 1 DUP BEGIN 13 < WHILE. «ESTO ES UN BUCLE QUE SE REPITE 12 VECES» CR 1+DUP REPEAT;
```

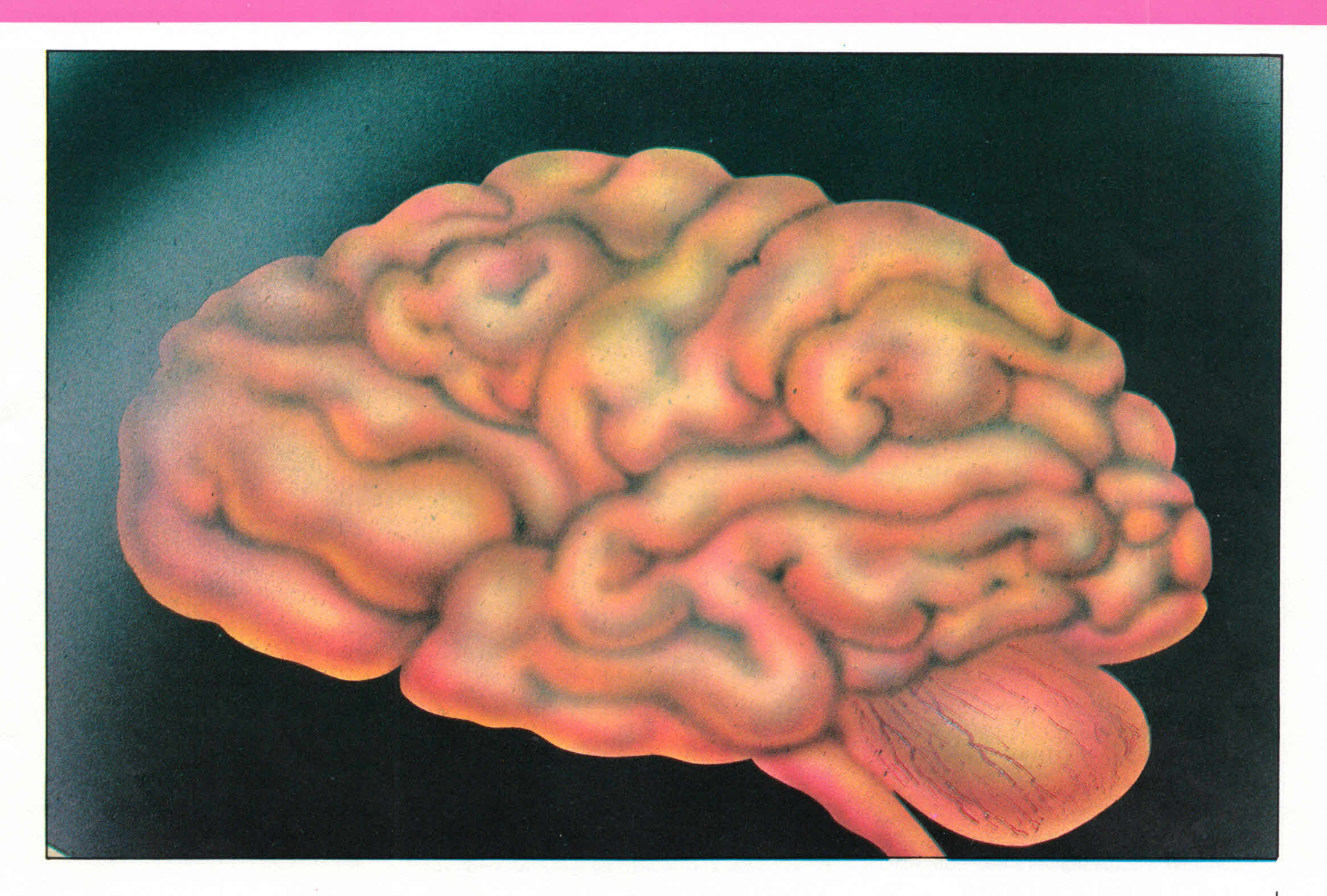
La segunda forma de bucle indefinido es la que hace que se esté ejecutando un proceso hasta que se cumpla una condición. Se trata de la instrucción GEGIN... UNTIL.

El formato más general de esta sentencia es:

BEGIN < acción > < condición > UNTIL

y su forma de trabajar es la siguiente:

BEGIN marca, como en el caso anterior, el comienzo del bucle. El cuerpo del mismo es



«acción» y será lo que se repita en el caso de no cumplirse la condición.

Después se evalúa la condición y deja en el stack un valor acorde con el resultado de la misma. UNTIL es la palabra clave que se encarga de encaminarnos a un sitio o a otro diferente. Cuando se encuentre con que el resultado lógico de la condición es falso *(cero)*, vuelve a BEGIN y repite el proceso.

Sin embargo, si es verdadero *(distinto de cero)* el programa continúa por lo que hay detrás de UNTIL.

Programa V

: BUCLE4 0 BEGIN. «ESTO ES UN BUCLE QUE SE REPITE 12 VECES» CR 1+DUP 12=UNTIL. «FIN»;

Estos dos últimos bucles funcionan de una manera muy parecida: se evalúa una condición y dependiendo del resultado se continúa fuera del mismo o se repite la ejecución de un determinado proceso.

Pero existe una gran diferencia entre ambos: el momento donde se comprueba si se cumple la condición.

En el primer caso realizamos el análisis antes de ejecutar el cuerpo del bucle, por tanto, puede darse el caso que la condición sea falsa y no ejecutemos acción alguna.

Sin embargo, en el segundo siempre se va a ejecutar el proceso que hayamos definido entre BEGIN y UNTIL (al menos una vez). Se debe esto a que ahora la evaluación de la condición se hace al final del mismo. Así que, ¡cuidado con este matiz!

Parece que se nos ha olvidado algo al hablar de este lenguaje: las variables. ¿Es que no se emplean?

La verdad es que sí, existen y se usan. Pero una de las cosas que da potencia y velocidad al Forth es precisamente que utiliza las variables solamente cuando es imprescindible hacerlo.

Para pasar parámetros a rutinas o a operadores empleamos el stack tal y como lo hemos venido haciendo hasta ahora ya que se trata de valores temporales y no tiene mucho interés almacenarlos continuamente.

Las variables en Forth

Pero quizá necesitemos alguna vez guardar un determinado dato permanentemente. Y para esto están las variables en éste y en la mayoría de los lenguajes. Pero en este caso estarán definidas a nivel global, es decir, que pueden ser utilizadas por cualquiera de las palabras que estén en el vocabulario.

La forma de declararlas es:

< valor inicial > VARIABLE < nombre >

y el nombre que le hayamos dado se incorpora directamente al vocabulario de variables. Con:

0 VARIABLE PUNTOS

estamos creando una variable que se llama PUNTOS, a la que damos un valor inicial cero.

Podemos dar, o asignar, un valor distinto del inicial a una variable, ya definida, a lo largo de un programa. La forma de hacerlo en Basic sería:

PUNTOS=100

pero en Forth no se hace así. Es un poco diferente, compárelo usted mismo:

100 PUNTOS!

Y también podemos hacer el proceso inverso, es decir, recuperar en el stack el valor que previamente habíamos asignado a una variable.

Para colocar el valor de PUNTOS en el elemento superior del stack bastaría con hacer:

PUNTOS

y después imprimirlo con la palabra «.».

Pero estas dos últimas instrucciones se podrían agrupar en una sola. Si escribimos:

PUNTOS ?

obtendríamos el mismo resultado.

Por si queremos conocer todas las variables que hay definidas hemos implementado un comando que consigue hacerlas aparecer en la pantalla. Tecleando:

*VARLIST

podremos obtener todo el vocabulario de variables incorporadas al lenguaje Forth. Intente definir unas cuantas y compruebe cómo funciona esta nueva sentencia.

Ahora nos surge una pregunta: ¿qué hacemos cuando ya esté creado un programa completo compuesto por varias palabras, que funcione y cumpla su cometido?

El Forth es un lenguaje como todos los demás, luego lo que debemos hacer es guardar el programa en cinta o en disco para así poder volver a utilizarlo cuantas veces queramos.

Si quiere almacenarlo, o salvarlo, escriba:

*SAVE < nombre >

y todas las palabras y variables definidas que estén en ese momento en la memoria pasarán a un soporte físico (disco o cinta) y de esta forma no las perderemos.

Para volver a cargarlas otra vez en la memoria siga el mismo procedimiento al que ya está acostumbrado en Basic:

*LOAD < nombre >

Después de dar un repaso general a unas cuantas palabras standard del Forth, visualice el diccionario básico mediante:

*VLIST

Observará que en él hay incluidas algunas palabras que no son propias, precisamente, del lenguaje tales como CLG, DRAW, MOVE, etc.

Se han añadido a esta versión para darle una mayor potencia, sobre todo a la hora de crear gráficos y dibujos en la pantalla. Se utilizan de una forma muy similar a como se hace en Basic, pero teniendo en cuenta la nueva notación invertida que hemos aprendido.

Como muestra de sus posibilidades le dejamos el Programa VI. Echele un vistazo e intente explicarse cómo funciona.

Programa VI

```
: CUADRADO —2 —2 MOVER DUP 0
DRAWR O SWAP DRAWR DUP MINUS 0
DRAWR DUP MINUS 0 SWAPT DRAWR;
: FIGURA 0CLS 200 320 MOVE 1 DO I
CUADRADO LOOP;
```

Y esto es todo por el momento. En una próxima oportunidad nos acercaremos un poco más a todas las palabras que utiliza este nuevo lenguaje Forth. Deseamos que, al menos, le llame un poquito la atención esta nueva forma de ver la programación.

```
10 REM ****************************
20 REM *        AMSTRAD FORTH      *
30 REM *                           *
40 REM *     MICROHOBBY AMSTRAD    *
50 REM ****************************
60 REM
70 REM INICIALIZACION
80 MODE 2:BORDER 13:INK 0,13:INK 1,
0
90 forth$=CHR$(12)+"Microhobby Fort
h V1.1"
100 OPENOUT "dummy":MEMORY HIMEM-1:
CLOSEOUT
110 temp=0:DEFINT a-z
120 DIM w$(130),p(130),beg(40),ff(4
0),df(40),bucle(40),ll(40),li(40)
130 er$(0)="OK":er$(1)="Desbordamie
nto de la capacidad minima del stac
k":er$(2)="Stack vacio"
140 er$(3)=" ya definido":er$(4)="
- nombre de variable ilegal":er$(5)
=" - Palabra no permitida":er$(6)="
Stack lleno":er$(7)="Stack de retor
no lleno"
150 spmax=100:DIM s(spmax):sp=-1
160 cvn=58:DIM cvoc$(cvn),dsp(cvn,1
)
170 FOR i=0 TO cvn:READ cvoc$(i),ds
p(i,0),dsp(i,1):NFXT
180 cvoc$(5)="."+CHR$(34)
190 umax=100:vmax=100:DIM uvoc$(uma
x),uvex$(umax),var$(vmax),var(vmax)
200 uvn=-1:vrn=-1:PRINT forth$:PRIN
T:PRINT er$(0)
210 ON ERROR GOTO 2500
220 REM ENTRADA DE COMANDOS
230 w$="":er=0:LINE INPUT ln$:IF ln
$="" THEN 660 ELSE IF LEN(ln$)>240
THEN PRINT "Linea demasiado larga":
GOTO 230
240 WHILE ASC(ln$)=32:IF LEN(ln$)>1
 THEN ln$=RIGHT$(ln$,LEN(ln$)-1):WE
ND ELSE 660
250 WHILE RIGHT$(ln$,1)=CHR$(32):ln
$=LEFT$(ln$,LEN(ln$)-1):WEND
260 ln$=UPPER$(ln$):IF ASC(ln$)=ASC
("*") THEN IF LEN(ln$)>1 AND LEFT$(
ln$,2)<>"* " THEN GOSUB 1670:IF er
THEN 230 ELSE w$="":GOTO 660
270 ln$=ln$+CHR$(32):x$="":q=1:wn=-
1:comp=0
280 WHILE q<LEN(ln$)
290 p=q:WHILE MID$(ln$,q,1)<>" ":q=
q+1:WEND
300 w$=MID$(ln$,p,q-p):IF w$=":" TH
EN IF wn=-1 AND RIGHT$(ln$,2)="; "
THEN comp=-1:GOTO 610 ELSE PRINT "D
efinicion incorrecta":GOTO 230
310 FOR i=cvn TO 0 STEP-1:IF cvoc$(
i)<>w$ THEN NEXT:GOTO 370
320 IF comp THEN IF wn=0 THEN er=3:
GOTO 660
330 IF wn>=0 THEN IF w$(wn)="VARIAB
LE" THEN er=4:GOTO 660
340 x$=x$+CHR$(0)+CHR$(i+14):IF w$<
>"."+CHR$(34) THEN 610
350 te=INSTR(q,ln$,CHR$(34)+CHR$(32
)):IF te=0 THEN PRINT".";CHR$(34);"
 sin ";CHR$(34):GOTO 230
360 x$=x$+MID$(ln$,q+1,te-q-1)+CHR$
(4):q=te+1:GOTO 610
370 FOR i=uvn TO 0 STEP-1:IF uvoc$(
i)<>w$ THEN NEXT:GOTO 410
380 IF comp THEN IF wn=0 THEN er=3:
GOTO 660
390 IF wn>=0 THEN IF w$(wn)="VARIAB
LE" THEN er=4:GOTO 660
400 x$=x$+CHR$(1)+CHR$(i+14):GOTO 6
10
410 FOR i=vrn TO 0 STEP -1:IF var$(
i)<>w$ THEN NEXT:GOTO 450
420 IF comp THEN IF wn=0 THEN er=3:
GOTO 660
430 IF wn>=0 THEN IF w$(wn)="VARIAB
LE" THEN er=3:GOTO 660
440 x$=x$+CHR$(2)+CHR$(i+14):GOTO 6
10
450 FOR i=1 TO LEN(w$)
460 IF i=1 AND (ASC(w$)=ASC("+") OR
 ASC(w$)=ASC("-")) AND LEN(w$)>1 TH
EN 480
470 IF MID$(w$,i,1)<"0" OR MID$(w$,
i,1)>"9" THEN 530
480 NEXT i
490 IF comp AND wn=0 THEN er=5:GOTO
 660
500 IF wn>=0 THEN IF w$(wn)="VARIAB
LE" THEN er=5:GOTO 660
510 IF VAL(w$)>32767 OR VAL(w$)<-32
767 THEN PRINT"Numero ";w$;" demasi
ado grande":GOTO 230
520 x$=x$+CHR$(3)+w$+CHR$(4):GOTO 6
10
530 IF wn<0 THEN 560 ELSE IF w$(wn)
<>"VARIABLE" THEN 560
540 IF comp AND w$=w$(1) THEN er=3:
GOTO 660
550 x$=x$+w$+CHR$(4):GOTO 610
560 IF w$<>";" THEN 580
570 IF comp=0 OR q<>LEN(ln$) THEN P
RINT"Punto y coma ilegal":GOTO 230
ELSE 610
580 IF comp THEN IF wn=0 THEN 610
590 IF comp THEN IF w$<>w$(1) THEN
er=5:GOTO 660 ELSE x$=x$+CHR$(1)+CH
R$(uvn+15):GOTO 610
600 er=5:GOTO 660
610 wn=wn+1:w$(wn)=w$
620 WHILE MID$(ln$,q,1)=" ":q=q+1:W
END
630 WEND
640 x$=x$+CHR$(13):IF comp THEN 690
 ELSE 730
650 REM RUTINA DE ERROR
660 IF POS(#0)>1 THEN PRINT CHR$(32
);
670 PRINT w$;er$(er):GOTO 230
680 REM COMPILAR NUEVA PALABRA
690 IF wn<3 THEN PRINT"Definicion i
ncompleta":GOTO 230
700 uvn=uvn+1:uvoc$(uvn)=w$(1):uvex
$(uvn)=x$
710 w$="":GOTO 660
720 REM EJECUCION DE COMANDOS
730 ln=0:w$(ln)=x$:er=0
740 GOSUB 750:w$="":GOTO 660
750 p(ln)=1:ff(ln)=0:df(ln)=0
760 WHILE MID$(w$(ln),p(ln),1)<>CHR
$(13)
770 clase=ASC(MID$(w$(ln),p(ln),1))
:p(ln)=p(ln)+1
780 IF clase<>0 THEN 890
790 pal=ASC(MID$(w$(ln),p(ln),1))-1
4:p(ln)=p(ln)+1
800 IF ff(ln)=0 OR pal=37 OR pal=39
 OR pal=40 THEN 830
810 IF pal=5 OR pal=32 THEN WHILE A
SC(MID$(w$(ln),p(ln),1))<>4:p(ln)=p
(ln)+1:WEND:p(ln)=p(ln)+1
820 GOTO 1030
830 IF sp+dsp(pal,1)<-1 OR sp+dsp(p
al,1)>spmax THEN er=1:GOTO 1040
840 IF sp-dsp(pal,0)<-1 THEN er=2:G
OTO 1040
850 sp=sp+dsp(pal,1)
860 IF pal<43 THEN ON pal+1 GOSUB 1
060,1080,1090,1100,1110,1120,1130,1
140,1150,1160,1170,1180,1190,1200,1
210,1220,1230,1240,1250,1260,1270,1
280,1290,1300,1310,1320,1330,1340,1
350,1360,1370,1380,1390,1400,1410,1
420,1430,1440,1450,1460,1470,1480,1
490
870 IF pal>42 THEN ON pal-42 GOSUB
1500,1510,1520,1530,1540,1550,1560,
1570,1580,1590,1600,1610,1620,1630,
1640,1650
880 IF er=0 THEN 1030 ELSE 1040
890 IF clase<>1 THEN 940
900 pal=ASC(MID$(w$(ln),p(ln),1))-1
4:p(ln)=p(ln)+1
910 IF ff(ln) THEN 1030
920 IF ln<34 THEN ln=ln+1:w$(ln)=uv
ex$(pal) ELSE er=7:RETURN
930 GOSUB 750:IF ln=0 OR er=0 THEN
1030 ELSE RETURN
940 IF clase<>2 THEN 980
950 pal=ASC(MID$(w$(ln),p(ln),1))-1
4:p(ln)=p(ln)+1
960 IF ff(ln) THEN 1030
970 sp=sp+1:s(sp)=@var(pal):GOTO 10
30
980 IF clase<>3 THEN er=1:GOTO 1040
990 p=p(ln):WHILE ASC(MID$(w$(ln),p
,1))<>4:p=p+1:WEND
1000 v=VAL(MID$(w$(ln),p(ln),p-p(ln
)+1)):p(ln)=p+1
```

```
1010 IF ff(ln) THEN 1030
1020 sp=sp+1:s(sp)=v
1030 WEND
1040 ln=ln-1:RETURN
1050 REM LISTA DE COMANDOS
1060 temp!=s(sp+1):IF temp!<0 THEN temp!=temp!+65536
1070 POKE s(sp+2),temp!-256*INT(temp!/256):POKE s(sp+2)+1,INT(temp!/256):RETURN
1080 s(sp)=s(sp)*s(sp+1):RETURN
1090 s(sp)=s(sp)+s(sp+1):RETURN
1100 s(sp)=s(sp)-s(sp+1):RETURN
1110 PRINT s(sp+1);CHR$(8);:RETURN
1120 WHILE ASC(MID$(w$(ln),p(ln),1))<>4 :PRINT MID$(w$(ln),p(ln),1);:p(ln)=p(ln)+1:WEND:p(ln)=p(ln)+1:RETURN
1130 s(sp)=INT(s(sp)/s(sp+1)):RETURN
1140 temp=s(sp):s(sp)=INT(s(sp-1)/s(sp)):s(sp-1)=s(sp-1)-s(sp)*temp:RETURN
1150 s(sp)=(s(sp)<0):RETURN
1160 s(sp)=(s(sp)=0):RETURN
1170 s(sp)=(s(sp)<s(sp+1)):RETURN
1180 s(sp)=(s(sp)=s(sp+1)):RETURN
1190 s(sp)=(s(sp)>s(sp+1)):RETURN
1200 temp!=PEEK(s(sp+1))+256*PEEK(s(sp+1)+1):IF temp!>32767 THEN temp!=temp!-65536:PRINT temp!;CHR$(8);:RETURN ELSE PRINT temp!;CHR$(8);:RETURN
1210 temp!=PEEK(s(sp))+256*PEEK(s(sp)+1):IF temp!>32767 THEN s(sp)=temp!-65536:RETURN ELSE s(sp)=temp!:RETURN
1220 s(sp)=ABS(s(sp)):RETURN
1230 s(sp)=s(sp) AND s(sp+1):RETURN
1240 s(sp)=PEEK(s(sp)):RETURN
1250 PRINT:RETURN
1260 RETURN
1270 s(sp)=s(sp-1):RETURN
1280 PRINT CHR$(s(sp+1));:RETURN
1290 in$=INKEY$:IF in$="" THEN 1290 ELSE s(sp)=ASC(in$):RETURN
1300 s(sp)=MAX(s(sp),s(sp+1)):RETURN
1310 s(sp)=MIN(s(sp),s(sp+1)):RETURN
1320 s(sp)=-s(sp):RETURN
1330 s(sp)=s(sp) MOD s(sp+1):RETURN
1340 s(sp)=s(sp) OR s(sp+1):RETURN
1350 s(sp)=s(sp-2):RETURN
1360 PRINT " ";:RETURN
1370 PRINT USING "&";SPACE$(s(sp+1)-256*INT(s(sp+1)/256));:RETURN
1380 temp=s(sp):s(sp)=s(sp-1):s(sp-1)=temp:RETURN
1390 vrn=vrn+1:var(vrn)=s(sp+1):WHILE ASC(MID$(w$(ln),p(ln),1))<>4:var$(vrn)=var$(vrn)+MID$(w$(ln),p(ln),1):p(ln)=p(ln)+1:WEND:p(ln)=p(ln)+1:RETURN
1400 s(sp)=s(sp) XOR s(sp+1):RETURN
1410 beg(ln)=p(ln):RETURN
1420 IF s(sp+1)=0 THEN p(ln)=beg(ln):RETURN ELSE RETURN
1430 IF s(sp+1)<>0 THEN RETURN ELSE ff(ln)=-1:RETURN
1440 IF ff(ln) THEN ff(ln)=0:RETURN ELSE p(ln)=beg(ln):RETURN
1450 IF s(sp+1)<>0 THEN RETURN ELSE ff(ln)=-1:RETURN
1460 ff(ln)=0:RETURN
1470 ff(ln)=-1-ff(ln):RETURN
1480 FOR i=0 TO uvn:uvoc$(i)="":uvex$(i)="":NEXT:uvn=-1:PRINT forth$:PRINT
1490 FOR i=0 TO vrn:var$(i)="":var(i)=0:NEXT:vrn=-1:RETURN
1500 temp=s(sp-2):s(sp-2)=s(sp-1):s(sp-1)=s(sp):s(sp)=temp:RETURN
1510 IF NOT df(ln) THEN df(ln)=-1:bucle(ln)=p(ln):ll(ln)=s(sp+1):li(ln)=s(sp+2):RETURN ELSE RETURN
1520 li(ln)=li(ln)+1:IF li(ln)<ll(ln) THEN p(ln)=bucle(ln):RETURN ELSE df(ln)=0:RETURN
1530 s(sp)=li(ln):RETURN
1540 CLG s(sp+1):RETURN
1550 DRAW s(sp+2),s(sp+1):RETURN
1560 DRAWR s(sp+2),s(sp+1):RETURN
1570 temp!=FRE(""):IF temp!>32767 THEN s(sp)=temp!-65536:RETURN ELSE s(sp)=temp!:RETURN
1580 MOVE s(sp+2),s(sp+1):RETURN
1590 MOVER s(sp+2),s(sp+1):RETURN
1600 PLOT s(sp+2),s(sp+1):RETURN
1610 PLOTR s(sp+2),s(sp+1):RETURN
1620 s(sp)=RND*32768:RETURN
1630 s(sp)=TEST(s(sp+1),s(sp)):RETURN
1640 s(sp)=TESTR(s(sp+1),s(sp)):RETURN
1650 x=XPOS:y=YPOS:PLOT 800,800,s(sp+1):MOVE x,y:RETURN
1660 REM PROCESO DE EDICION DE COMANDOS
1670 er=0:w$=""
1680 IF ln$="*VLIST" THEN FOR i=cvn TO 0 STEP -1: PRINT cvoc$(i),:NEXT:PRINT:RETURN
1690 IF ln$<>"*LIST" THEN 1750
1700 FOR i=uvn TO 0 STEP -1
1710 PRINT uvoc$(i),
1720 IF INKEY$="" THEN 1740
1730 WHILE INKEY$="":WEND
1740 NEXT:PRINT:RETURN
1750 IF LEFT$(ln$,6)<>"*LIST " THEN 2010
1760 w$=RIGHT$(ln$,LEN(ln$)-6)
1770 WHILE ASC(w$)=32:w$=RIGHT$(w$,LEN(w$)-1):WEND
1780 FOR i=uvn TO 0 STEP -1:IF w$<>uvoc$(i) THEN NEXT:PRINT w$;" - Palabra desconocida":er=1:RETURN
1790 x$=uvex$(i)
1800 WHILE ASC(x$)<>13
1810 clase=ASC(x$):x$=RIGHT$(x$,LEN(x$)-1)
1820 IF clase<>0 THEN 1890
1830 pal=ASC(x$)-14:x$=RIGHT$(x$,LEN(x$)-1)
1840 PRINT cvoc$(pal);" ";
1850 IF pal<>5 AND pal<>32 THEN 1990
1860 WHILE ASC(x$)<>4:PRINT LEFT$(x$,1);:x$=RIGHT$(x$,LEN(x$)-1):WEND
1870 x$=RIGHT$(x$,LEN(x$)-1):IF pal=5 AND clase<>3 THEN PRINT CHR$(34);
1880 PRINT " ";:GOTO 1990
1890 IF clase<>1 THEN 1930
1900 pal=ASC(x$)-14:x$=RIGHT$(x$,LEN(x$)-1)
1910 PRINT uvoc$(pal);" ";
1920 GOTO 1990
1930 IF clase<>2 THEN 1970
1940 pal=ASC(x$)-14:x$=RIGHT$(x$,LEN(x$)-1)
1950 PRINT var$(pal);" ";
1960 GOTO 1990
1970 IF clase<>3 THEN PRINT er$(1):GOTO 2000
1980 GOTO 1860
1990 WEND
2000 PRINT:RETURN
2010 IF LEFT$(ln$,8)<>"*FORGET " THEN 2100
2020 w$=RIGHT$(ln$,LEN(ln$)-8)
2030 WHILE ASC(w$)=32:w$=RIGHT$(w$,LEN(w$)-1):WEND
2040 FOR i=uvn TO 0 STEP -1:IF w$<>uvoc$(i) THEN NEXT:GOTO 2070
2050 FOR j=i TO uvn:uvoc$(j)="":exec$(j)="":NEXT
2060 uvn=i-1:RETURN
2070 FOR i=vrn TO 0 STEP -1:IF w$<>var$(i) THEN NEXT:PRINT w$;" - Palabra desconocida":er=-1:RETURN
2080 FOR j=i TO vrn-1:var$(j)=var$(j+1):var(j)=var(j+1):NFXT
2090 vrn=vrn-1:RETURN
2100 IF ln$="*SAVE" THEN 2440
2110 IF LEFT$(ln$,6)<>"*SAVE " THEN 2260
2120 w$=RIGHT$(ln$,LEN(ln$)-6)
2130 WHILE ASC(w$)=32:w$=RIGHT$(w$,LEN(w$)-1):WEND
2140 dp=INSTR(w$,".")
2150 IF dp=0 THEN w1$=w$:w2$="4TH" ELSE w1$=LEFT$(w$,dp-1):w2$=RIGHT$(w$,LEN(w$)-dp)
2160 IF w1$="" OR LEN(w1$)>8 OR LEN(w2$)>3 THEN 2440
2170 IF w2$="" THEN w2$="4TH"
2180 w$=w1$+"."+w2$
2190 OPENOUT w$
2200 PRINT #9,uvn:PRINT #9,vrn
2210 FOR i=0 TO uvn:PRINT #9,uvoc$(i):PRINT #9,LEN(uvex$(i))
2220 FOR j=1 TO LEN(uvex$(i)):PRINT #9,ASC(MID$(uvex$(i),j,1));:NEXT j
2230 NEXT i
2240 FOR i=0 TO vrn:PRINT #9,var$(i):PRINT #9,var(i):NEXT
2250 CLOSEOUT:RETURN
2260 IF ln$="*LOAD" THEN 2440
2270 IF LEFT$(ln$,6)<>"*LOAD " THEN 2440
2280 w$=RIGHT$(ln$,LEN(ln$)-6)
2290 WHILE ASC(w$)=32:w$=RIGHT$(w$,LEN(w$)-1):WEND
2300 dp=INSTR(w$,".")
2310 IF dp=0 THEN w1$=w$:w2$="4TH" ELSE w1$=LEFT$(w$,dp-1):w2$=RIGHT$(w$,LEN(w$)-dp)
2320 IF w1$="" OR LEN(w1$)>8 OR LEN(w2$)>3 THEN 2440
2330 IF w2$="" THEN w2$="4TH"
2340 w$=w1$+"."+w2$
2350 OPENIN w$
2360 INPUT #9,uvn:INPUT #9,vrn
2370 ERASE uvoc$,uvex$,var$,var
2380 DIM uvoc$(umax),uvex$(umax),var$(vmax),var(vmax)
2390 FOR i=0 TO uvn:INPUT #9,uvoc$(i):INPUT #9,lux
2400 FOR j=1 TO lux:INPUT #9,temp:uvex$(i)=uvex$(i)+CHR$(temp):NEXT j
2410 NEXT i
2420 FOR i=0 TO vrn:INPUT #9,var$(i):INPUT #9,var(i):NEXT
2430 CLOSEIN:RETURN
2440 IF ln$<>"*VARLIST" THEN PRINT "Comando desconocido o incompleto":er=-1:RETURN
2450 FOR i=vrn TO 0 STEP -1
2460 w$=var$(i)+CHR$(32)+STR$(var(i))+SPACE$(2):PRINT w$,
2470 IF INKEY$="" THEN 2490
2480 WHILE INKEY$="":WEND
2490 NEXT:PRINT:RETURN
2500 er=6:IF sp>100 THEN sp=100:RESUME NEXT ELSE w$="":RESUME 660
2510 REM DATOS DE PALABRAS RESERVADAS
2520 DATA "!",2,-2,"*",2,-1,"+",2,-1,"-",2,-1,".",0,-1,"?",0,0,"/",2,-1,"/MOD",2,0,"0<",1,0,"0=",1,0,"<",2,-1
2530 DATA "=",2,-1,">",2,-1,"?",1,-1,"@",1,0,"ABS",1,0,"AND",2,-1,"C@",1,0,"CR",0,0,"DROP",1,-1,"DUP",1,1
2540 DATA "EMIT",1,-1,"KEY",0,1,"MAX",2,-1,"MIN",2,-1,"MINUS",1,0,"MOD",2,-1,"OR",2,-1,"OVER",2,1
2550 DATA "SPACE",0,0,"SPACES",1,-1,"SWAP",2,0,"VARIABLE",1,-1,"XOR",2,-1,"BEGIN",0,0,"UNTIL",1,-1
2560 DATA "WHILE",1,-1,"REPEAT",0,0,"IF",1,-1,"THEN",0,0,"ELSE",0,0,"FORTH",0,0,"CLEAR",0,0,"ROT",3,0,"DO",2,-2,"LOOP",0,0,"I",0,+1
2570 REM DATOS PARA PALABRAS AMSTRAD
2580 DATA "CLG",1,-1,"DRAW",2,-2,"DRAWR",2,-2,"FRE",0,1,"MOVE",2,-2,"MOVER",2,-2
2590 DATA "PLOT",2,-2,"PLOTR",2,-2,"RND",0,1,"TEST",2,1,"TESTR",2,1,"GRAPEN",1,-1
```



Para que tus dedos no realicen el trabajo duro, M.H. AMSTRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicítanoslo.

EUROPA

El software educativo, por desgracia, no tiene un papel protagonista dentro de la inmensa cantidad de programas que roldean a los ordenadores Amstrad.

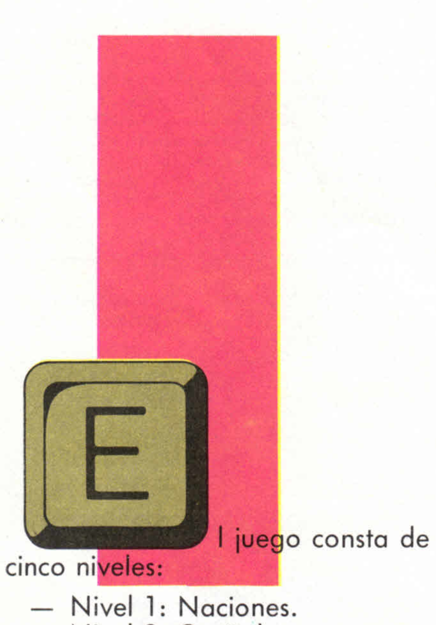
Tal vez porque sea muy difícil de hacer, tal vez porque los que comercializan programas no están muy seguros del mercado potencial en este campo, o quizás porque en nuestro país la enseñanza asistida por ordenador está en «mantillas».

El caso es que siempre resulta agradable ver que un lector se preocupa por explorar un terreno considerablemente virgen, contando con que los medios de que dispone son inferiores, con mucho, a los de cualquier casa de soft.

El programa «Europa» es una buena idea para aprender geografía en varios sentidos a la vez. Su presentación, a mi juicio, es impecable, y se nota que el programa está hecho con amor, cuidando al máximo el detalle.

Alguien podría objetar que «Europa» es excesivamente largo y tedioso de teclear, y que no hubiera estado mal emplear un algoritmo en lugar de dibujar el mapa de Europa punto a punto.

Bien, no hubiera estado mal, pero hay programas escritos con la cabeza y programas escritos con el corazón; yo creo que «Europa» es de los segundos, y apuesto por él.



El juego consta de cinco niveles:

— Nivel 1: Naciones.
— Nivel 2: Capitales.
— Nivel 3: Naciones-capitales.
— Nivel 4: Organización militar.
— Nivel 5: Organización económica.

En el primer nivel aparecerá el mapa de Europa y un cuadrado rojo en una nación europea.

El ordenador te preguntará qué nación es donde se encuentra el cuadrado rojo. Si adivinas la nación te da 20 puntos y te pregunta otra nación.

Cuanto tengas 40 puntos, pulsando «L» cambias de nivel.

En el segundo nivel, lo que hay que poner es la capital de la nación. Cuando tengas 80 puntos, pasas a otro nivel pulsando **«L».**

... y así sucesivamente con **«Naciones-capitales»**, **«Organizaciones militares»** y **«Organizaciones económicas».**

Los cambios de nivel se producen a:

— los 40 pts;
— los 80 pts;
— los 120 pts;
— los 160 pts;
— los 200 pts.

A los 200 puntos vuelves a comenzar.

Variables

E Valor **«x»** del cuadrado en pantalla de gráficos.
F Valor **«y»** del cuadrado en pantalla de gráficos.
H Valor del incremento.
I Color del cuadrado (Siempre rojo 11)
Q$ Nombre de la nación, capital, etc.
BC$, BD$, BE$, BF$, BG$, BH$, BI$, BJ$, BK$, BL$, BLL$, BM$, BN$, BO$, BP$, BQ$, BR$, BS$, BT$, BU$, BW$, BX$, BY$, Datos.
Ca Nivel.
K$ Nombre de la nación, capital, etc. que introduces.
LL Puntuación.
AZ Número de países ya apelados.

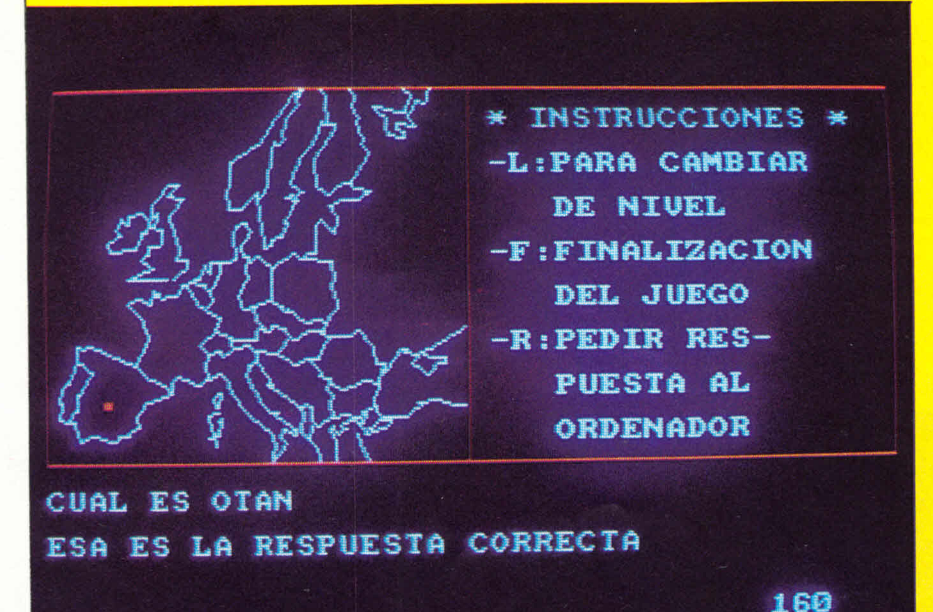
Instrucciones

— Simplemente hacer lo que el ordenador diga, tiene las instrucciones incorporadas.

— Si cuando pones el nombre de la nación, capital, etc. escribes **«L»** y tu puntuación es 40, 80, 120, 160 ó 200, cambias de nivel.

— Si escribes **«F»** finaliza el juego.

— Si escribes **«R»** te da la respuesta el ordenador.






```
10 KEY 1,"PACTO DE VARSOVIA"
20 REM
30 REM ****************************
****
40 REM ****  CARLOS REYES MUNCLUS
****
50 REM ****************************
****
60 REM
70 INK 0,0:BORDER 0:INK 1,11:INK 2,
6:INK 3,13
80 PEN 2:CLS:PRINT:PRINT TAB(3)"* I
NSTRUCCIONES DEL PRIMER NIVEL *"
90 PEN 1:PRINT:PRINT:PRINT"-Tienes
que poner el nombre de la ":PRINT:P
RINT:PRINT"nacion donde se encuentr
e el cuadrado ":PRINT:PRINT:PRINT"r
ojo"
100 PRINT:PRINT:PRINT"Cualquier fal
ta ortografica la":PRINT:PRINT:PRIN
T"interpreto como error"
110 REM
120 REM *********** INICIO ********
*
130 REM
140 XC$=INKEY$
150 IF XC$="" THEN GOTO 140
160 CLS
170 I=5
180 INK 1,0:INK 2,0:INK 3,0
190 PRINT:PEN 3:PRINT TAB(22)"* INS
TRUCCIONES *"
200 PEN 1:PRINT:PRINT TAB(22)"-L:PA
RA CAMBIAR"
210 PRINT:PRINT TAB(25)"DE NIVEL"
220 PRINT:PRINT TAB(22)"-F:FINALIZA
CION"
230 PRINT:PRINT TAB(25)"DEL JUEGO"
240 PRINT:PRINT TAB(22)"-R:PEDIR RE
S-"
250 PRINT:PRINT TAB(25)"PUESTA AL"

260 PRINT:PRINT TAB(25)"ORDENADOR"
270 REM
280 REM *********** DIBUJO ********
*
290 REM
300 PLOT 195,399,1:DRAW 192,393
310 PLOT 192,393:DRAW 190,394
320 PLOT 190,394:DRAW 180,383
330 PLOT 180,383:DRAW 165,358
340 PLOT 165,358:DRAW 143,347
350 PLOT 143,347:DRAW 142,328
360 PLOT 142,328:DRAW 140,326
370 PLOT 140,326:DRAW 142,313
380 PLOT 142,313:DRAW 153,313
390 PLOT 153,313:DRAW 164,325
400 PLOT 164,325:DRAW 171,285
410 PLOT 171,285:DRAW 177,287
420 PLOT 177,287:DRAW 176,290
430 PLOT 176,290:DRAW 185,292
440 PLOT 185,292:DRAW 188,302
450 PLOT 188,302:DRAW 187,303
460 PLOT 187,303:DRAW 193,315
470 PLOT 193,315:DRAW 197,318
480 PLOT 197,318:DRAW 187,320
490 PLOT 187,320:DRAW 193,322
500 PLOT 193,322:DRAW 195,320
510 PLOT 195,320:DRAW 200,324
520 PLOT 200,324:DRAW 191,331
530 PLOT 191,331:DRAW 196,349
540 PLOT 196,349:DRAW 207,358
550 PLOT 207,358:DRAW 212,374
560 PLOT 212,374:DRAW 217,373
570 PLOT 217,373:DRAW 215,393
580 PLOT 215,393:DRAW 210,399
590 PLOT 197,399:DRAW 193,388
600 PLOT 193,388:DRAW 194,387
610 PLOT 194,387:DRAW 185,377
620 PLOT 185,377:DRAW 182,365
630 PLOT 182,365:DRAW 184,363
640 PLOT 184,363:DRAW 181,359
650 PLOT 181,359:DRAW 177,361
660 PLOT 177,361:DRAW 174,354
670 PLOT 174,354:DRAW 172,329
680 PLOT 172,329:DRAW 165,319
690 PLOT 217,373:DRAW 222,372
700 PLOT 222,372:DRAW 222,368
710 PLOT 222,368:DRAW 209,349
720 PLOT 209,349:DRAW 210,329
730 PLOT 210,329:DRAW 220,323
740 PLOT 220,323:DRAW 230,330
750 PLOT 230,330:DRAW 236,330
760 PLOT 236,330:DRAW 239,326
770 PLOT 239,326:DRAW 243,327
780 PLOT 243,327:DRAW 250,324
790 PLOT 250,324:DRAW 242,319
800 PLOT 242,319:DRAW 230,318
810 PLOT 236,330:DRAW 249,353
820 PLOT 249,353:DRAW 243,358
830 PLOT 243,358:DRAW 244,362
840 PLOT 244,362:DRAW 240,368
850 PLOT 240,368:DRAW 239,375
860 PLOT 239,375:DRAW 240,376
870 PLOT 240,376:DRAW 235,385
880 PLOT 235,385:DRAW 237,392
890 PLOT 237,392:DRAW 232,397
900 PLOT 232,398:DRAW 232,399
910 PLOT 270,399:DRAW 279,392
920 PLOT 279,392:DRAW 275,382
930 PLOT 275,381:DRAW 273,381
940 PLOT 273,381:DRAW 250,385
950 PLOT 250,385:DRAW 260,376
960 PLOT 260,376:DRAW 260,373
970 PLOT 260,373:DRAW 263,365
980 PLOT 263,365:DRAW 274,363
990 PLOT 274,363:DRAW 275,366
1000 PLOT 275,366:DRAW 268,369
1010 PLOT 268,369:DRAW 269,373
1020 PLOT 269,373:DRAW 283,371
1030 PLOT 283,371:DRAW 276,377
1040 PLOT 276,377:DRAW 283,388
1050 PLOT 283,388:DRAW 289,387
1060 PLOT 289,387:DRAW 290,388
1070 PLOT 290,388:DRAW 290,395
1080 PLOT 290,395:DRAW 295,399
1090 PLOT 230,318:DRAW 221,314
1100 PLOT 221,314:DRAW 226,307
1110 PLOT 226,307:DRAW 225,295
1120 PLOT 225,295:DRAW 214,304
1130 PLOT 214,304:DRAW 210,300
1140 PLOT 210,300:DRAW 210,281
1150 PLOT 210,281:DRAW 206,281
1160 PLOT 206,281:DRAW 200,276
1170 PLOT 200,276:DRAW 223,274
1180 PLOT 223,274:DRAW 225,260
1190 PLOT 225,260:DRAW 223,259
1200 PLOT 223,259:DRAW 227,243
1210 PLOT 227,243:DRAW 218,230
1220 PLOT 218,230:DRAW 215,223
1230 PLOT 215,223:DRAW 220,220
1240 PLOT 220,220:DRAW 237,220
1250 PLOT 237,220:DRAW 242,225
1260 PLOT 242,225:DRAW 252,213
1270 PLOT 252,213:DRAW 254,200
1280 PLOT 254,200:DRAW 261,200
1290 PLOT 261,200:DRAW 261,195
1300 PLOT 261,195:DRAW 273,211
1310 PLOT 273,211:DRAW 276,211
1320 PLOT 276,211:DRAW 275,211
1330 PLOT 275,208:DRAW 285,206
1340 PLOT 285,206:DRAW 281,200
1350 PLOT 281,200:DRAW 293,193
1360 PLOT 293,193:DRAW 299,200
1370 PLOT 299,200:DRAW 304,201
1380 PLOT 304,201:DRAW 305,205
1390 PLOT 305,205:DRAW 295,204
1400 PLOT 295,204:DRAW 287,209
1410 PLOT 287,209:DRAW 293,210
1420 PLOT 293,210:DRAW 320,227
1430 PLOT 261,195:DRAW 258,197
1440 PLOT 258,197:DRAW 256,186
1450 PLOT 256,186:DRAW 253,175
1460 PLOT 253,175:DRAW 255,170
1470 PLOT 255,170:DRAW 254,168
1480 PLOT 254,168:DRAW 259,163
1490 PLOT 259,163:DRAW 277,162
1500 PLOT 277,162:DRAW 289,172
1510 PLOT 284,172:DRAW 299,175
1520 PLOT 299,175:DRAW 307,173
1530 PLOT 307,173:DRAW 310,170
1540 PLOT 310,170:DRAW 320,170
1550 PLOT 255,170:DRAW 245,167
1560 PLOT 245,167:DRAW 246,166
1570 PLOT 246,166:DRAW 244,159
1580 PLOT 244,159:DRAW 243,156
1590 PLOT 243,156:DRAW 245,155
1600 PLOT 245,155:DRAW 246,152
1610 PLOT 246,152:DRAW 258,160
1620 PLOT 258,160:DRAW 265,160
1630 PLOT 265,160:DRAW 260,154
1640 PLOT 260,154:DRAW 246,150
1650 PLOT 241,150:DRAW 248,130
1660 PLOT 244,159:DRAW 236,160
1670 PLOT 236,160:DRAW 229,157
1680 PLOT 229,157:DRAW 232,154
1690 PLOT 232,154:DRAW 226,150
1700 PLOT 226,150:DRAW 222,156
1710 PLOT 222,156:DRAW 220,154
1720 PLOT 220,154:DRAW 225,144
1730 PLOT 225,144:DRAW 225,130
1740 PLOT 245,167:DRAW 243,163
1750 PLOT 243,163:DRAW 239,162
1760 PLOT 239,162:DRAW 233,165
1770 PLOT 233,165:DRAW 223,163
1780 PLOT 223,163:DRAW 211,157
1790 PLOT 206,157:DRAW 205,148
1800 PLOT 205.148:DRAW 212,137
```

```
1810 PLOT 212,137:DRAW 213,130
1820 PLOT 205,148:DRAW 199,153
1830 PLOT 199,153:DRAW 200,166
1840 PLOT 200,166:DRAW 198,170
1850 PLOT 198,170:DRAW 173,187
1860 PLOT 173,187:DRAW 177,187
1870 PLOT 177,187:DRAW 169,198
1880 PLOT 169,198:DRAW 165,203
1890 PLOT 165,203:DRAW 158,200
1900 PLOT 158,200:DRAW 159,197
1910 PLOT 159,197:DRAW 156,194
1920 PLOT 156,194:DRAW 167,175
1930 PLOT 167,175:DRAW 175,167
1940 PLOT 175,167:DRAW 180,167
1950 PLOT 180,167:DRAW 178,165
1960 PLOT 178,165:DRAW 195,154
1970 PLOT 195,154:DRAW 194,150
1980 PLOT 194,150:DRAW 185,155
1990 PLOT 185,155:DRAW 182,150
2000 PLOT 182,150:DRAW 186,145
2010 PLOT 186,145:DRAW 181,136
2020 PLOT 181,136:DRAW 176,133
2030 PLOT 176,133:DRAW 176,136
2040 PLOT 176,136:DRAW 179,142
2050 PLOT 179,142:DRAW 176,150
2060 PLOT 176,150:DRAW 165,161
2070 PLOT 165,161:DRAW 160,162
2080 PLOT 160,162:DRAW 145,180
2090 PLOT 145,180:DRAW 143,189
2100 PLOT 143,189:DRAW 136,195
2110 PLOT 136,195:DRAW 126,190
2120 PLOT 126,190:DRAW 121,186
2130 PLOT 121,186:DRAW 103,192
2140 PLOT 103,192:DRAW 98,188
2150 PLOT 98,188:DRAW 98,180
2160 PLOT 98,180:DRAW 95,178
2170 PLOT 95,178:DRAW 78,174
2180 PLOT 78,174:DRAW 70,164
2190 PLOT 70,164:DRAW 73,157
2200 PLOT 73,157:DRAW 65,153
2210 PLOT 65,153:DRAW 65,150
2220 PLOT 65,150:DRAW 57,150
2230 PLOT 57,150:DRAW 55,146
2240 PLOT 55,146:DRAW 36,150
2250 PLOT 36,150:DRAW 26,147
2260 PLOT 26,147:DRAW 23,160
2270 PLOT 23,160:DRAW 18,160
2280 PLOT 18,160:DRAW 10,163
2290 PLOT 10,163:DRAW 13,175
2300 PLOT 13,175:DRAW 10,175
2310 PLOT 10,175:DRAW 12,184
2320 PLOT 12,184:DRAW 19,189
2330 PLOT 19,189:DRAW 24,204
2340 PLOT 24,204:DRAW 25,215
2350 PLOT 25,215:DRAW 31,213
2360 PLOT 31,213:DRAW 35,215
2370 PLOT 35,215:DRAW 71,200
2380 PLOT 71,200:DRAW 80,190
2390 PLOT 80,190:DRAW 84,191
2400 PLOT 84,191:DRAW 98,183
2410 PLOT 18,160:DRAW 21,166
2420 PLOT 21,166:DRAW 24,168
2430 PLOT 24,168:DRAW 25,175
2440 PLOT 25,175:DRAW 25,180
2450 PLOT 25,180:DRAW 27,180
2460 PLOT 27,180:DRAW 37,197
2470 PLOT 37,197:DRAW 24,204
2480 PLOT 71,200:DRAW 80,220
2490 PLOT 80,220:DRAW 76,232
2500 PLOT 76,232:DRAW 65,244
2510 PLOT 65,244:DRAW 73,249
2520 PLOT 73,249:DRAW 82,244
2530 PLOT 82,244:DRAW 84,254
2540 PLOT 84,254:DRAW 88,253
2550 PLOT 88,253:DRAW 86,250
2560 PLOT 86,250:DRAW 93,248
2570 PLOT 93,248:DRAW 95,250
2580 PLOT 95,250:DRAW 103,250
2590 PLOT 103,250:DRAW 106,258
2600 PLOT 106,258:DRAW 110,259
2610 PLOT 110,259:DRAW 113,260
2620 PLOT 113,260:DRAW 120,262
2630 PLOT 120,262:DRAW 125,273
2640 PLOT 125,273:DRAW 129,267
2650 PLOT 129,267:DRAW 129,275
2660 PLOT 129,275:DRAW 137,275
2670 PLOT 137,275:DRAW 145,274
2680 PLOT 145,274:DRAW 149,277
2690 PLOT 149,277:DRAW 147,294
2700 PLOT 147,294:DRAW 149,300
2710 PLOT 149,300:DRAW 156,303
2720 PLOT 156,303:DRAW 160,297
2730 PLOT 160,297:DRAW 153,287
2740 PLOT 153,287:DRAW 155,279
2750 PLOT 155,279:DRAW 159,279
2760 PLOT 159,279:DRAW 160,274
2770 PLOT 160,274:DRAW 167,278
2780 PLOT 167,278:DRAW 177,270
2790 PLOT 177,270:DRAW 195,280
2800 PLOT 195,280:DRAW 200,276
2810 PLOT 177,270:DRAW 177,246
2820 PLOT 177,246:DRAW 185,243
2830 PLOT 185,243:DRAW 185,240
2840 PLOT 185,240:DRAW 188,238
2850 PLOT 188,238:DRAW 188,241
2860 PLOT 188,241:DRAW 200,233
2870 PLOT 200,233:DRAW 204,231
2880 PLOT 204,231:DRAW 212,232
2890 PLOT 212,232:DRAW 218,230
2900 PLOT 110,259:DRAW 121,243
```




```
2910 PLOT 121,243:DRAW 136,234
2920 PLOT 136,234:DRAW 132,224
2930 PLOT 132,224:DRAW 121,215
2940 PLOT 121,215:DRAW 127,208
2950 PLOT 127,208:DRAW 124,201
2960 PLOT 124,201:DRAW 126,190
2970 PLOT 131,176:DRAW 137,180
2980 PLOT 137,180:DRAW 135,170
2990 PLOT 135,170:DRAW 132,170
3000 PLOT 132,170:DRAW 131,176
3010 PLOT 127,164:DRAW 135,166
3020 PLOT 135,160:DRAW 137,162
3030 PLOT 137,162:DRAW 135,150
3040 PLOT 135,150:DRAW 127,148
3050 PLOT 133,142:DRAW 127,164
3060 PLOT 113,260:DRAW 124,258
3070 PLOT 124,258:DRAW 129,253
3080 PLOT 129,253:DRAW 130,247
3090 PLOT 130,247:DRAW 125,240
3100 PLOT 129,253:DRAW 137,268
3110 PLOT 137,268:DRAW 137,275
3120 PLOT 159,279:DRAW 157,268
3130 PLOT 157,268:DRAW 159,265
3140 PLOT 159,265:DRAW 151,250
3150 PLOT 151,250:DRAW 152,244
3160 PLOT 152,244:DRAW 155,241
3170 PLOT 155,241:DRAW 177,246
3180 PLOT 132,224:DRAW 150,218
3190 PLOT 150,218:DRAW 164,218
3200 PLOT 164,218:DRAW 164,225
3210 PLOT 170,227:DRAW 165,233
3220 PLOT 165,233:DRAW 162,242
3230 PLOT 127,208:DRAW 137,210
3240 PLOT 137,210:DRAW 139,206
3250 PLOT 139,206:DRAW 149,212
3260 PLOT 149,212:DRAW 145,215
3270 PLOT 145,215:DRAW 145,220
3280 PLOT 149,212:DRAW 158,214
3290 PLOT 158,214:DRAW 165,210
3300 PLOT 165,210:DRAW 165,203
3310 PLOT 165,210:DRAW 181,209
3320 PLOT 181,209:DRAW 190,220
3330 PLOT 190,220:DRAW 187,226
3340 PLOT 187,226:DRAW 177,229
3350 PLOT 177,229:DRAW 175,225
3360 PLOT 175,225:DRAW 167,226
3370 PLOT 190,220:DRAW 195,218
3380 PLOT 195,218:DRAW 207,224
3390 PLOT 207,224:DRAW 215,224
3400 PLOT 211,157:DRAW 207,162
3410 PLOT 207,162:DRAW 207,171
3420 PLOT 207,171:DRAW 203,173
3430 PLOT 203,173:DRAW 198,169
3440 PLOT 223,163:DRAW 220,174
3450 PLOT 220,174:DRAW 223,178
3460 PLOT 223,178:DRAW 219,185
3470 PLOT 219,185:DRAW 221,186
3480 PLOT 221,186:DRAW 230,183
3490 PLOT 230,183:DRAW 249,188
3500 PLOT 249,188:DRAW 256,186
3510 PLOT 221,186:DRAW 220,191
3520 PLOT 220,191:DRAW 212,192
3530 PLOT 212,192:DRAW 214,195
3540 PLOT 214,195:DRAW 205,204
3550 PLOT 205,204:DRAW 195,200
3560 PLOT 195,200:DRAW 181,209
3570 PLOT 205,204:DRAW 212,205
3580 PLOT 212,205:DRAW 216,215
3590 PLOT 216,215:DRAW 220,220
3600 PLOT 100,263:DRAW 110,270
3610 PLOT 110,270:DRAW 110,276
3620 PLOT 110,276:DRAW 103,277
3630 PLOT 103,277:DRAW 105,285
3640 PLOT 105,285:DRAW 96,308
3650 PLOT 96,308:DRAW 106,321
3660 PLOT 106,321:DRAW 95,322
3670 PLOT 95,322:DRAW 103,330
3680 PLOT 103,330:DRAW 94,331
3690 PLOT 94,331:DRAW 83,317
3700 PLOT 83,317:DRAW 83,307
3710 PLOT 83,307:DRAW 87,310
3720 PLOT 87,310:DRAW 86,305
3730 PLOT 86,305:DRAW 82,301
3740 PLOT 82,301:DRAW 90,298
3750 PLOT 90,298:DRAW 90,290
3760 PLOT 90,290:DRAW 88,286
3770 PLOT 88,286:DRAW 81,287
3780 PLOT 81,287:DRAW 81,281
3790 PLOT 81,281:DRAW 74,276
3800 PLOT 74,276:DRAW 85,270
3810 PLOT 85,270:DRAW 65,262
3820 PLOT 65,262:DRAW 95,263
3830 PLOT 48,288:DRAW 59,295
3840 PLOT 59,295:DRAW 55,298
3850 PLOT 55,298:DRAW 58,307
3860 PLOT 58,307:DRAW 65,303
3870 PLOT 65,303:DRAW 65,308
3880 PLOT 65,308:DRAW 75,310
3890 PLOT 75,310:DRAW 80,305
3900 PLOT 80,305:DRAW 79,297
3910 PLOT 79,297:DRAW 75,296
3920 PLOT 75,296:DRAW 71,285
3930 PLOT 71,285:DRAW 51,282
3940 PLOT 51,282:DRAW 48,288
3950 PLOT 1,399,2:DRAW 320,399
3960 PLOT 320,399:DRAW 320,130
3970 PLOT 320,130:DRAW 1,130
3980 PLOT 1,130:DRAW 1,399
3990 PLOT 320,399,2:DRAW 639,399
4000 PLOT 639,399:DRAW 639,130
4010 PLOT 639,130:DRAW 320,130
4020 INK 1,11:INK 2,6:INK 3,13
4030 PEN 3:PRINT:PRINT:B=25
4040 GOTO 6180
4050 REM
4060 REM ********* RAMDOM *********
4070 REM
4080 E=45:F=170:G=5:H=5:I=11
4090 Q$=BC$
4100 R=R+1
4110 IF R>=2 THEN GOTO 6460
4120 GOSUB 5870
4130 GOTO 5920
4140 E=100:F=215:G=5:H=5:I=11
4150 Q$=BD$
4160 T=T+1
4170 IF T>=2 THEN GOTO 6460
4180 GOSUB 5870
4190 GOTO 5920
4200 E=65:F=290:G=5:H=5:I=11
4210 Q$=BE$
4220 U=U+1
4230 IF U>=2 THEN GOTO 6460
4240 GOSUB 5870
4250 GOTO 5920
4260 E=90:F=270:G=5:H=5:I=11
4270 Q$=BF$
4280 W=W+1
4290 IF W>=2 THEN GOTO 6460
4300 GOSUB 5870
4310 GOTO 5920
4320 E=160:F=165:G=5:H=5:I=11
4330 Q$=BG$
4340 X=X+1
4350 IF X>=2 THEN GOTO 6460
4360 GOSUB 5870
4370 GOTO 5920
4380 E=135:F=212:G=5:H=5:I=11
4390 Q$=BH$
```

```
4400 Y=Y+1
4410 IF Y>=2 THEN GOTO 6460
4420 GOSUB 5870
4430 GOTO 5920
4440 AB=AB+1
4450 IF AB>=2 THEN GOTO 6460
4460 E=117:F=250:G=5:H=5:I=11:GOSU
 5870
4470 Q$=BI$
4480 GOTO 5920
4490 E=15:F=175:G=5:H=5:I=11
4500 Q$=BJ$
4510 Z=Z+1
4520 IF Z>=2 THEN GOTO 6460
4530 GOSUB 5870
4540 GOTO 5920
4550 E=125:F=260:G=5:H=5:I=11
4560 Q$=BK$
4570 AA=AA+1
4580 IF AA>=2 THEN GOTO 6460
4590 GOSUB 5870
4600 GOTO 5920
4610 E=140:F=240:G=5:H=5:I=11
4620 Q$=BL$
4630 AC=AC+1
4640 IF AC>=2 THEN GOTO 6460
4650 GOSUB 5870
4660 GOTO 5920
4670 E=160:F=250:G=5:H=5:I=11
4680 Q$=BLL$
4690 AD=AD+1
4700 IF AD>=2 THEN GOTO 6460
4710 GOSUB 5870
4720 GOTO 5920
4730 E=170:F=215:G=5:H=5:I=11
4740 Q$=BM$
4750 AE=AE+1
4760 IF AE>=2 THEN GOTO 6460
4770 GOSUB 5870
4780 GOTO 5920
4790 E=190:F=185:G=5:H=5:I=11
4800 Q$=BN$
4810 AF=AF+1
4820 IF AF>=2 THEN GOTO 6460
4830 GOSUB 5870
4840 GOTO 5920
4850 E=175:F=235:G=5:H=5:I=11
4860 Q$=BO$
4870 AG=AG+1
4880 IF AG>=2 THEN GOTO 6460
4890 GOSUB 5870
4900 GOTO 5920
4910 E=200:F=255:G=5:H=5:I=11
4920 Q$=BP$
4930 AH=AH+1
4940 IF AH>=2 THEN GOTO 6460
4950 GOSUB 5870
4960 GOTO 5920
4970 E=230:F=200:G=5:H=5:I=11
4980 Q$=BQ$
4990 AI=AI+1
5000 IF AI>=2 THEN GOTO 6460
5010 GOSUB 5870
5020 GOTO 5920
5030 E=200:F=210:G=5:H=5:I=11
5040 Q$=BR$
5050 AJ=AJ+1
5060 IF AJ>=2 THEN GOTO 6460
5070 GOSUB 5870
5080 GOTO 5920
5090 E=200:F=155:G=5:H=5:I=11
5100 Q$=BS$
5110 AK=AK+1
5120 IF AK>=2 THEN GOTO 6460
5130 GOSUB 5870
5140 GOTO 5920
5150 E=215:F=135:G=5:H=5:I=11
5160 Q$=BT$
5170 AL=AL+1
5180 IF AL>=2 THEN GOTO 6460
5190 GOSUB 5870
5200 GOTO 5920
5210 E=235:F=170:G=5:H=5:I=11
5220 Q$=BU$
5230 AM=AM+1
5240 IF AM>=2 THEN GOTO 6460
5250 GOSUB 5870
5260 GOTO 5920
5270 E=150:F=330:G=5:H=5:I=11
5280 Q$=BV$
5290 AN=AN+1
5300 IF AN>=2 THEN GOTO 6460
5310 GOSUB 5870
5320 GOTO 5920
5330 E=175:F=305:G=5:H=5:I=11
5340 Q$=BW$
5350 AO=AO+1
5360 IF AO>=2 THEN GOTO 6460
5370 GOSUB 5870
5380 GOTO 5920
5390 E=225:F=350:G=5:H=5:I=11
5400 Q$=BX$
5410 AP=AP+1
5420 IF AP>=2 THEN GOTO 6460
5430 GOSUB 5870
5440 GOTO 5920
5450 E=260:F=260:G=5:H=5:I=11
5460 Q$=BY$
5470 AQ=AQ+1
5480 IF AQ>=2 THEN GOTO 6460
5490 GOSUB 5870
5500 GOTO 5920
5510 IF L>=40 THEN GOTO 5530
5520 GOTO 6810
5530 R=0:T=0:U=0:W=0:X=0:Y=0:Z=0:AB
=0:AA=0:AC=0:AD=0:AE=0:AF=0:AG=0:AH
=0:AI=0:AJ=0:AK=0:AL=0:AM=0:AN=0:AO
=0:AP=0:AQ=0
5540 IF CA=1 THEN GOTO 5620
5550 IF CA=2 THEN GOTO 5680
5560 IF CA=3 THEN GOTO 5740
5570 IF CA=4 THEN GOTO 5790
5580 IF CA=5 THEN GOTO 5430
5590 REM
5600 REM ********** NIVELES *******
***
5610 REM
5620 PEN 2:CLS:PRINT:PRINT TAB(2)"*
 INSTRUCCIONES DEL SEGUNDO NIVEL *"
5630 PEN 1:PRINT:PRINT:PRINT"-Tiene
s que poner el nombre de la":PRINT:
PRINT:PRINT"capital donde se encuen
tre el cuadrado":PRINT:PRINT:PRINT"
rojo":PRINT:PRINT:PRINT"cualquier f
alta ortografica la ":PRINT:PRINT:P
RINT"interpreto como error"
5640 ZS$=INKEY$
5650 IF ZS$="" THEN GOTO 5640
5660 CA=2:CC$="CAPITAL"
5670 GOTO 160
5680 PEN 2:CLS:PRINT:PRINT TAB(2)"*
 INSTRUCIONES DEL TERCER NIVEL":PEN
```

* INSTRUCCIONES DEL CUARTO NIVEL *

Tienes que poner el nombre de
la organizacion militar de la nacion
donde se encuentre el cuadrado rojo
Cualquier falta ortografica
la interpreto como error

```
 1:PRINT:PRINT:PRINT"-Tienes que po
ner el nombre de la":PRINT:PRINT:PR
INT"nacion-capital donde se encuent
re el":PRINT:PRINT:PRINT"cuadrado r
ojo"
5690 PRINT:PRINT:PRINT"cualquier fa
lta ortografica la":PRINT:PRINT:PRI
NT"interpreto como error"
5700 ZX$=INKEY$
5710 IF ZX$="" THEN GOTO 5700
5720 CA=3
5730 GOTO 160
5740 CLS:PRINT:PEN 3:PRINT TAB(2)"*
 INSTRUCCIONES DEL CUARTO NIVEL *":
PRINT:PRINT:PEN 1:PRINT"Tienes que
poner el nombre de":PRINT:PRINT"la
organizacion militar de la nacion":
PRINT:PRINT"donde se encuentre el c
uadrado rojo"
5750 PRINT:PRINT"Cualquier falta or
tografica":PRINT:PRINT"la interpret
o como error"
```




```
5760 CA=4
5770 PS$=INKEY$:IF PS$="" THEN GOTO
 5770
5780 GOTO 160
5790 CLS:PRINT:PEN 3:PRINT TAB(2)"*
 INSTRUCCIONES DEL QUINTO NIVEL *":
PRINT:PRINT:PEN 1:PRINT "Tienes que
 poner el nombre de":PRINT:PRINT"la
 organizacion politica de la nacion
":PRINT:PRINT"donde se encuentre el
 cuadrado rojo"
5800 PRINT:PRINT"Cualquier falta or
tografica":PRINT:PRINT"la interpret
o como error"
5810 CA=5
5820 CR$=INKEY$:IF CR$="" THEN GOTO
 5820
5830 GOTO 160
5840 REM
5850 REM ***** PINTAR CUADRADO ****
5860 REM
5870 FOR J=1 TO G
5880 PLOT E,F,2:DRAW E+H,F
5890 F=F+1
5900 NEXT J
5910 RETURN
5920 REM
5930 REM ******** PREGUNTA ********
5940 REM
5950 LOCATE 1,19:INPUT"CUAL ES ",K$

5960 AZ=AZ+1
5970 IF K$="L" THEN GOTO 6010
5980 IF K$="F" THEN GOTO 6090
5990 IF K$="R" THEN GOTO 6100
6000 GOTO 6120
6010 IF LL=40 THEN GOTO 5620
6020 IF LL=80 THEN GOTO 5680
6030 IF LL=120 THEN GOTO 5740
6040 IF LL=160 THEN GOTO 5790
6050 IF LL=200 THEN GOTO 6810

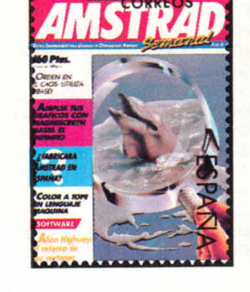
6060 LOCATE 1,19:PRINT"PUEDES CAMBI
AR DE NIVEL CUANDO TU ":PRINT:PRINT
 "PUNTUACION SEA 40,80,120 o 160":F
OR HJK=1 TO 2000:NEXT:LOCATE 1,19:P
RINT"
        ":LOCATE 1,21:PRINT"
                                ":LOC
ATE 1,19
6070 R=R-1
6080 GOTO 5920
6090 CLS:END
6100 PEN 1:PRINT:PRINT Q$:FOR S=1 T
O 2500:NEXT S:IF AZ=23 THEN GOTO 55
10
6110 GOTO 6740
6120 IF K$=Q$ THEN GOTO 6140
6130 IF K$<>Q$ THEN GOTO 6160
6140 PRINT:PRINT"ESA ES LA RESPUEST
A CORRECTA":PEN 1:SOUND 1,100,2,7:L
L=LL+20:PRINT:PRINT:PRINT TAB(34) L
L:PEN 3:FOR LLA=1 TO 2500:NEXT:IF L
L>=200 THEN GOTO 6820:IF AZ=23 THEN
 GOTO 5510
6150 GOTO 6740
6160 PRINT:PRINT"NO,LA RESPUESTA CO
RRECTA ES":PEN 1:SOUND 1,800,25,7:P
RINT:PRINT Q$:PEN 1:PRINT TAB (34)
LL:PEN 3:INK 0,0:BORDER 0:FOR V=1 T
O 2500:NEXT V:INK 0,0:BORDER 0:IF A
Z=23 THEN GOTO 5510
6170 GOTO 6740
6180 IF CA=1 THEN GOTO 6260
6190 IF CA=2 THEN GOTO 6290
6200 IF CA=3 THEN GOTO 6320
6210 IF CA=4 THEN GOTO 6390
6220 IF CA=5 THEN GOTO 6420
```

```
6230 REM
6240 REM *********** DATOS **********
**
6250 REM
6260 BC$="ESPANA":BD$="FRANCIA":BE$
="IRLANDA":BF$="INGLATERRA":BG$="IT
ALIA":BH$="SUIZA":BI$="BELGICA":BJ$
="PORTUGAL":BK$="HOLANDA":BL$="ALEM
ANIA OCCIDENTAL":BLL$="ALEMANIA ORI
ENTAL":BM$="AUSTRIA":BN$="YUGOSLAVI
A":BO$="CHECOSLOVAQUIA":BP$="POLONI
A"
6270 BQ$="RUMANIA":BR$="HUNGRIA":BS
$="ALBANIA":BT$="GRECIA":BU$="BULGA
RIA":BV$="NORUEGA":BW$="SUECIA":BX$
="FINLANDIA":BY$="RUSIA"
6280 GOTO 6460
6290 BC$="MADID":BD$="PARIS":BE$="D
UBLIN":BF$="LONDRES":BG$="ROMA":BH$
="BERNA":BI$="BRUSELAS":BJ$="LISBOA
":BK$="AMSTERDAM":BL$="BONN":BLL$="
BERLIN":BM$="VIENA"
6300 BN$="BELGRADO":BO$="PRAGA":BP$
="VARSOVIA":BQ$="BUCAREST":BR$="BUD
APEST":BS$="TIRANA"
6310 BT$="ATENAS":BU$="SOFIA":BV$="
OSLO":BW$="ESTOCOLMO":BX$="HELSINKI
":BY$="MOSCU":GOTO 6460
6320 BC$="ESPANA-MADRID":BD$="FRANC
IA-PARIS":BE$="IRLANDA-DUBLIN":BF$=
"INGLATERRA-LONDRES":BG$="ITALIA-RO
MA":BH$="SUIZA-BERNA":BI$="BELGICA-
BRUSELAS":BJ$="PORTUGAL-LISBOA":BK$
="HOLANDA-AMSTERDAM":BL$="ALEMANIA
OCCIDENTAL-BONN"
6330 BLL$="ALEMANIA ORIENTAL-BERLIN
":BM$="AUSTRIA-VIENA"
6340 BP$="POLONIA-VARSOVIA":BQ$="RU
MANIA-BUCAREST"
6350 BR$="HUNGRIA-BUDAPEST":BS$="AL
BANIA-TIRANA"
6360 BN$="YUGOSLAVIA-BELGRADO":BO$=
"CHECOSLOVAQUIA-PRAGA"
6370 BT$="GRECIA-ATENAS":BU$="BULGA
RIA-SOFIA":BV$="NORUEGA-OSLO":BW$="
SUECIA-ESTOCOLMO":BX$="FINLANDIA-HE
LSINKI":BY$="RUSIA-MOSCU"
6380 GOTO 6460
6390 BC$="OTAN":BD$="OTAN":BE$="NUL
O":BF$="OTAN":BG$="OTAN":BH$="NULO"
:BI$="OTAN":BJ$="OTAN":BK$="OTAN":B
L$="NULO":BLL$="PACTO DE VARSOVIA":
BM$="NULO":BN$="NULO"
6400 BO$="PACTO DE VARSOVIA":BP$="P
ACTO DE VARSOVIA":BQ$="PACTO DE VAR
SOVIA":BR$="PACTO DE VARSOVIA":BS$=
"PACTO DE VARSOVIA":BT$="NULO":BU$=
"PACTO DE VARSOVIA":BV$="OTAN":BW$=
"NULO":BX$="NULO":BY$="PACTO DE VAR
SOVIA"

6410 GOTO 6460
6420 BC$="CEE":BD$="CEE":BE$="CEE":
BF$="CEE":BG$="CEE":BH$="NULO":BI$=
"CEE":BJ$="CEE":BK$="CEE":BL$="CEE"
:BM$="NULO":BN$="NULO":BO$="COMECON
":BP$="COMECON":BQ$="COMECON":BR$="
COMECON":BS$="NULO":BT$="CEE":BU$="
COMECON"
6430 BV$="NULO":BW$="NULO":BX$="NUL
O":BY$="COMECON"
6440 GOTO 6460
6450 REM
6460 A=INT(RND(1)*B)
6470 IF A=1 THEN GOTO 4080
6480 IF A=2 THEN GOTO 4140
6490 IF A=3 THEN GOTO 4200
6500 IF A=4 THEN GOTO 4260
6510 IF A=5 THEN GOTO 4320
6520 IF A=6 THEN GOTO 4380
6530 IF A=7 THEN GOTO 4440
6540 IF A=8 THEN GOTO 4490
6550 IF A=9 THEN GOTO 4550
6560 IF A=10 THEN GOTO 4610
6570 IF A=11 THEN GOTO 4670
6580 IF A=12 THEN GOTO 4730
6590 IF A=13 THEN GOTO 4790
6600 IF A=14 THEN GOTO 4850
6610 IF A=15 THEN GOTO 4910
6620 IF A=16 THEN GOTO 4970
6630 IF A=17 THEN GOTO 5030
6640 IF A=18 THEN GOTO 5090
6650 IF A=19 THEN GOTO 5150
6660 IF A=20 THEN GOTO 5210
6670 IF A=21 THEN GOTO 5270
6680 IF A=22 THEN GOTO 5330
6690 IF A=23 THEN GOTO 5390
6700 IF A=24 THEN GOTO 5450
6710 REM
6720 REM ************ RND *********
**
6730 REM
6740 FOR DE=1 TO G
6750 PLOT E,F-5,0:DRAW E+H,F-5
6760 F=F+1
6770 NEXT
6780 LOCATE 1,19:PRINT"
6790 GOTO 4040
6800 GOTO 6460
6810 CLS:END
6820 CLS:RUN
```



Correo..., más rápido...



Con el fin de acelerar lo más posible el **correo**, y poder resolver o contestar a todas las dudas y sugerencias que llegan a nuestra redacción, a partir de esta semana os rogamos, en beneficio de todos, consignar en el sobre, en lugar bien visible, una de las denominaciones siguientes:

— **Suscripciones AMSTRAD.** Para todos aquellos casos relacionados con petición de cintas, números atrasados, formalización de suscripciones, devoluciones etc...

— **Mercado Común AMSTRAD.** Compras, ventas, intercambios, clubs...

— **Sin duda alguna AMSTRAD.** Para que nos enviéis todas vuestras dudas.

— **Serie Oro AMSTRAD.** Para los programas que nos enviéis para su publicación.

— **Sugerencias AMSTRAD.** Para vuestras críticas, sugerencias o cualquier opinión que queráis vertir sobre la revista.



ORDEMANIA SOFT
por fin..
GESPACK
Paquete integrado de gestión que le permite a Vd., de forma fácil y optimizando el tiempo, la gestión global de su empresa.
Este paquete incorpora los programas de CONTABILIDAD, FACTURACION y CONTROL DE STOCKS además de un programa de CONTROL DE PEDIDOS y todo de forma interactiva. Controle su empresa por sólo 29.900 pts.
Contabilidad
Contabilidad de fácil manejo y de gran potencia que permite trabajar con cuentas de hasta cuatro niveles, con capacidad según diskette de 500/1.000 cuentas y de 2.000/10.000 asientos.
Permite modificar o dar de baja apuntes ya integrados en el Mayor, programaciones de cierres, ficheros de Contabilidad y Cuenta de Explotación, ejecución de balances comparativos, reenlazándolos por meses, clave acceso restringido, etc...
19.900 ptas.
Facturación
Programa de gran agilidad y rapidez que incorpora el Control de Clientes, con gran capacidad de datos, Artículos, Albaranes, Facturas y recibos.
Generación automática y manual de documentos, valoración, a voluntad de los albaranes, todo tipo de listado, incluido el del IVA de las facturas emitidas para la declaración de Hacienda, etc...
15.500 ptas.
Control de stocks
Gran capacidad de datos, le permitirán a Vd. llevar con claridad y sencillez el control de su stock.
El programa le permite llevar un libro de entradas/salidas, reorganizarlo, hacer listados de stocks..., le avisará de los límites de stocks, mínimo y máximo por artículo, etc...
Todo para la llevanza de su almacén.
14.900 ptas.
disponibles para: PCW 8256
PCW 8512
Asimismo, Contabilidad disponible para CPC 664/6128 9.900 ptas.
OFITES INFORMATICA
DISTRIBUIDOR EXCLUSIVO DE ORDEMANIA SOFT
• IVA NO INCLUIDO
Condiciones especiales para distribuidores
Si tiene alguna dificultad en obtenerlos diríjase a
Ofites Informática
Avda. Isabel II, 16 - 8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

CONTROL DEL CRT

Pero, ¿qué es el CRT? Pues, sencillamente, las iniciales de «Cathode Ray Tube», que traducido a nuestro idioma quiere decir, literalmente, «Tubo de Rayos Catódicos», o en términos menos técnicos, la pantalla «física» del monitor de su Amstrad, que también es un tubo de rayos catódicos como el de un televisor cualquiera.



De lo anterior se deduce que el artículo que presentamos está dedicado a cómo controlar la pantalla. Y efectivamente, así es, pero desde un punto de vista distinto al habitual del BASIC y, sin embargo, sin utilizar código máquina ni PEEK's ni POKE's, a menos que, por razones de velocidad de ejecución, se decida emplear otros medios más rápidos que sólo son posibles desde código máquina, y para los que la base teórica que expondremos seguirá siendo igualmente válida.

Con las órdenes existentes en BASIC podemos hacer un montón de cosas **«EN la pantalla»,** pero también hay un comando que nos permitirá hacer otras tantas «CON la pantalla». Y dicho comando no es, ni más ni menos, que el comando **«OUT».**

EN LA PANTALLA Y CON LA PANTALLA

El comando OUT generalmente pasa desapercibido para el programador normal, sobre todo porque en los manuales no se explica cuál puede ser su utilidad práctica, limitándose a dar en dos líneas una escueta descripción de la acción que realiza y nada más. Transcribimos para aquéllos que no tengan el manual a mano:

«OUT < número de portal >, < expresión entera >
Envía el valor del parámetro entero al portal de SALIDA correspondiente a la dirección mencionada.»

Aun cuando tan sucinta definición es totalmente exacta, lo menos que puede ocurrir es que nos deje absolutamente indiferentes si no se complementa con un ejemplo o con una explicación de lo que es un portal de salida.

Intentaremos dar un poco de luz sobre el tema a lo largo del artículo, pero antes, y para desperezar los dedos, teclearemos en modo directo las siguientes instrucciones, operando en **MODE 1** y procurando no equivocarse:

OUT &BC00, 1 [ENTER] OUT &BD00, 20 [ENTER]

¿Qué es lo que ha ocurrido? Aparentemente, se han insertado líneas en blanco entre las que acaba de escribir, pero intente escribir algo, de más de 20 caracteres, o moverse con el cursor más allá de la columna 20 y verá cómo el cursor salta a la primera columna de la línea siguiente. ¡EUREKA! Pensará que hemos descubierto el comando WINDOW, sólo que más complicado. PUES NO. Puede comprobarlo tecleando MODE 1, con lo que volveríamos a la pantalla normal, pero que en esta ocasión no sirve de nada. Para ver lo que ha sucedido teclee:

BORDER 15 [ENTER]

Esta orden pone el borde la pantalla de color naranja y, en estas circunstancias, además, la mitad derecha de la pantalla también aparecerá de este color, lo que quiere decir que la pantalla útil se ha reducido a la mitad. Para ser más exactos, los 40 caracteres de una línea normal ahora se distribuyen en dos líneas de 20 caracteres, es decir, la pantalla de 25×40=1.000 caracteres ahora se ha transformado en 50 líneas de 20 caracteres. Imagínese la pantalla como una masa de letras que comprimimos horizontalmente: las letras al no poder salirse de la pantalla, se desplazarán hacia abajo, reestructurando la visualización, pero sin perderse ni una, solamente que no las vemos. Compruébelo desplazando el cursor por debajo de la última línea visible, pulse ESCAPE y siga bajando el cursor hasta que empiece el scroll hacia arriba, con lo que aparecerá el «*Break*» que no veíamos antes.

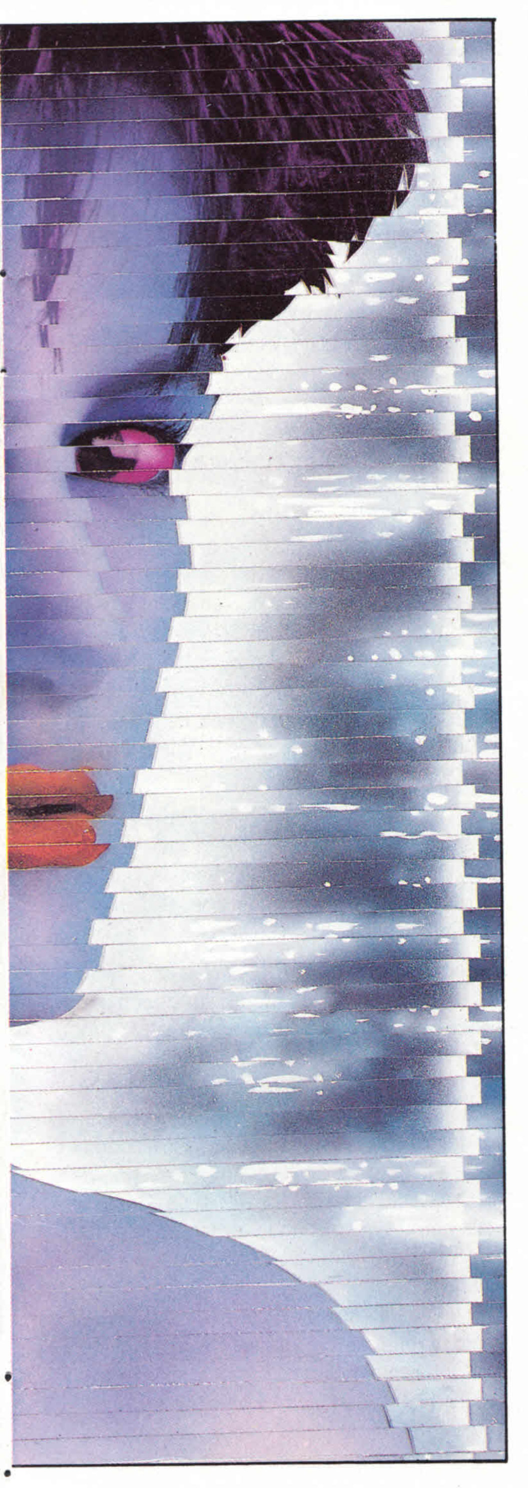
Para volver a la pantalla normal escriba:

OUT &BC00,1: OUT &BD00,40 [ENTER]

o resetee el ordenador.

Ahora se comprenderá la diferencia entre hacer cosas «EN la pantalla» y «CON la pantalla», puesto que la orden OUT afectará a toda la pantalla y no a zonas de la misma.

Antes de que se le ocurra empezar a experimentar por su cuenta haciendo OUT's a diestro y siniestro, estamos en la obligación moral de advertir que un OUT de cualquier valor en cualquier portal de salida, sin un ligero



conocimiento de lo que se hace, puede producir un «hermoso» bloqueo del sistema, siendo necesario desconectar el ordenador, y si no pruebe con:

OUT &BC00,0: OUT &BD00,1

UN POCO DE TEORIA

Todos los ordenadores están compuestos físicamente por una serie de elementos que nos permiten trabajar con él. El elemento más importante es el microprocesador o Unidad de Control de Proceso (CPU), que en los **Amstrads** es el renombrado Z80A, y que no es más que una *pastillita* de silicio de unos cuantos milímetros cuadrados conteniendo cerca de 8.000 transistores, interconectados de tal manera que posibilitan hacer cosas tales como jugar a marcianitos o llevar la contabilidad de una empresa.

Pero decimos que «posibilitan» porque, a pesar de la potencia del microprocesador, éste por sí solo no es incapaz de hacer absolutamente nada si no se encuentra conectado con otros dispositivos como un teclado, una pantalla o la memoria; mediante los cuales los podemos dar órdenes o recibir los resultados de una serie de operaciones o almacenar datos, respectivamente.

El Z80A es el cerebro que controla, directa o indirectamente, todas las acciones que realiza el ordenador, y entre ellas se encuentran: leer el teclado, manejar la pantalla, generar sonidos, controlar el cassette, enviar datos a la impresora, leer y escribir en la memoria, etc.

Nos centraremos en el área del control de la pantalla. El Z80 maneja la pantalla indirectamente a través de un chip o circuito integrado llamado CRTC, cuyas siglas corresponden a **«Cathode Ray Tube Controller»** (Controlador del Tubo de Rayos Catódicos).

El CRTC realiza las funciones de mantenimiento de la visualización en pantalla, controlando las dimensiones y la posición de la zona de exposición de información en la superficie del Tubo, y generando las direcciones de memoria donde están los datos de pantalla, entre otras cosas.

Todo esto se consigue utilizando el contenido de los 18 registros de que dispone el CRTC. Podemos entender un registro como un dispositivo cuya misión es retener una información para posteriormente ser tratada por el CRTC. Cada uno de estos registros puede contener un número de un byte de longitud, esto es, con un valor entre cero y 255.

Modificando el contenido de los registros del CRTC modificaremos las características de la pantalla, según hemos podido comprobar. ¿Y cómo modificar un registro? Nada más sencillo, con la orden **OUT**.

Como ya hemos dicho, el CRTC es un circuito conectado al microprocesador, y éste se comunica con aquél a través de unos canales que se encuentran en los llamados *PORTALES* de SALIDA. Un PORTAL no es más que una dirección de memoria donde se depositan datos para ser enviados al dispositivo correspondiente a ese número de portal. Explicándolo de otra forma, hacer un OUT de un Número a un Portal es como enviar una carta con un mensaje a una persona. El OUT es la acción de enviar la carta; el Número sería el mensaje, y el Portal sería la dirección de esa persona.

Necesitamos, pues, conocer la dirección del portal de salida correspondiente al Controlador del Tubo de Rayos Catódicos para poder enviarle los datos que queramos.

PROGRAMACCION

El CRTC tiene dos portales o direcciones, cada uno de los cuales tiene una misión específica. El primer portal se utiliza para indicarle al CRTC el número del registro que queremos seleccionar. Y el segundo portal es por donde le enviaremos el valor al registro previamente seleccionado. Las direcciones son:
&BC00: para seleccionar número de registro.
&BD00: para modificar contenido del registro.

Aun cuando hemos dicho que el CRTC tiene 18 registros, el sistema sólo permite acceder a los primeros 16, por lo que el número que tendremos que enviar al primer portal del CRTC, el &BC00, será un número entero entre cero y 15, inclusive, por aquello de que los ordenadores empiezan a contar desde cero. Lo haremos del siguiente modo:

OUT &BC00, <num. reg> siendo num. reg> un valor entre 0 y 15.

para modificar el contenido del registro seleccionado:

OUT &BD00, <dato> siendo <dato> un valor entre 0 y 255.

En la tabla 1 damos los nombres de cada uno de los registros, así como los valores estándar y los valores mínimos y máximos en cuya gama no se bloqueara el ordenador. Insistimos en que cualquier valor fuera de esta gama nos obligará a tener que desconectar nuestro **Amstrad.**

También existen portales de ENTRADA al microprocesador, donde recoger datos mediante la orden del BASIC **IN**, pero no los trataremos por salirse del tema que abordamos. Unicamente decir al respecto que un uso inapropiado de las entradas puede ocasionar **daños físicos** a los circuitos del **ordenador**.

Comentar todos y cada uno de los efectos que se producen al modificar los registros del CRTC con los distintos valores posibles es prácticamente imposible. Por otra parte, enviando OUT's en modo «directo» y de forma desordenada nos encontraremos con que es una forma un tanto pesada de investigar, y otras veces habremos cambiado tantos registros que ya no sabremos cómo volver a la pantalla normal.

Por todo ello, creemos que la mejor solución está en facilitar las cosas con un programa que nos permita manejar los registros a nuestro antojo, pudiendo recuperar la pantalla normal cuando queramos y ver qué registros hemos modificado hasta ese momento, para, de esta manera, sacar nuestras propias conclusiones de cómo inciden los diferentes parámetros enviados al CRTC, sobre todo, pensando en el dicho de que una imagen vale más que mil palabras. Todo esto nos lo permitirá el programa **«CONTROL DEL CRT»** que luego comentaremos.

FUNCIONAMIENTO DEL PROGRAMA «Control del CRT»

TABLA 1. REGISTROS DEL CRTC

N. Reg	Estand.	Min.	Max.	Nombre del Registro
0	63	46	100	Total
1	40	0	64	Horizontal Visualizado
2	46	0	63	Posición de sincronismo horizontal
3	142	(*)	(*)	Ancho del sincronismo
4	38	0	255	Total Vertical
5	0	0	255	Ajuste del Total Vertical
6	25	0	50	Vertical Visualizado
7	30	0	255	Posición de sincronismo vertical
8	30	0	255	Modo solapado y sesgado
9	7	0	255	Máximo número de líneas de barrido
10	0	0	255	Comienzo del cursor sintetizado
11	0	0	255	Fin del cursor sintetizado
12	48	0	255	Dirección comienzo (byte alto) rastreo memoria de pantalla
13	0	0	255	Idem (byte bajo)
14	192	0	255	Registro del cursor (byte alto)
15	0	0	255	Idem (byte bajo)

(*) = Dependiente del contexto de los demás registros.

Volviendo sobre el famoso dicho, pruebe a teclear el siguiente programa que nos muestra una forma más vistosa de presentar las pantallas en nuestros programas:

```
10 BORDER 15: MODE 0
20 FOR a=40 TO 0 STEP-1: GOSUB 60: NEXT: '—— Recoge Pantalla
30 GOSUB 70: '——Pintar Pantalla
40 FOR a=0 TO 40: GOSUB 60: NEXT: '——Extiende Pantalla
50 CALL &BB06: GOTO 20: '——Espera pulsación y vuelve a empezar
60 CALL &BD19: OUT &BC00,1: OUT &BD00,a: RETURN: 'Ejecuta OUT's
70 t=TIME
80 CLG RND*13
90 WHILE TIME <t+300
100 DRAW 640*RND, 400*RND, 13*RND: TAG: PRINT CHR$ (RND*26+64);
110 WEND
120 RETURN
```

A los que posean el juego **BEACH HEAD** les resultará familiar esta forma de recoger la pantalla. Está basada en variar el contenido del registro 1 del CRTC entre los valores 40 y 0, mediante un bucle FOR... NEXT STEP —1, para recoger la pantalla, y al revés, entre 0 y 40, para extenderla de nuevo.

Antes de cambiar el registro con los OUT's correspondientes, hacemos una llamada al FIRMWARE, en &BD19, para sincronizar el cambio que se va a realizar con el haz de barrido de pantalla, consiguiéndose una mayor suavidad en el movimiento. Es lo mismo que hacer un FRAME en el CPC 6128. Para ver la necesidad del CALL &BD19, pruebe a quitarlo.

Otro de los ejemplos que se me ocurren como consecuencia de indagar con los registros del CRTC utilizando el programa «CONTROL CRT» es el siguiente:

```
10 FOR d=1 TO 3: READ men$
20 men$=SPACE$ ((40-LEN (men$))/2)+ +men$: '——Centra mensaje
30 MODE 1
40 LOCATE 1,20
50 FOR a=1 TO 40
60 letra$=MID$ (men$,a,1)
70 GOSUB 140: '——Ejecuta OUT's
80 PRINT letra$;
90 NEXT
100 FOR r=0 to 1500: NEXT: '——Retardo
110 LOCATE 1,25: PRINT STRING$ (20,10): '——Sube mensaje
120 NEXT
130 RUN
140 CALL &BD19: OUT &BC00,13: OUT &BD00,a: RETURN
150 DATA MICROHOBBY AMSTRAD ESPECIAL
160 DATA Presenta:
170 DATA *CONTROL DEL CRT*
```

Esta vez se modifica el registro 13, que contiene el byte bajo de la dirección de comienzo de rastreo de la memoria de pantalla, por lo que, al ir desplazando dicha dirección con el comando OUT, se desplaza a su vez la pantalla, produciendo un suave **SCROLL** horizontal de la pantalla.

Esta última técnica, en combinación con unos cuantos conocimientos de código máquina para obtener la rapidez necesaria, nos permitiría construir un programa que utilizara la pantalla de su **Amstrad** como un tablón de anuncios electrónico a todo color, como ya he podido ver en algunos sitios.

Antes de hacerlo correr es conveniente salvarlo en cinta/disco por si hubiera algún OUT perdido que nos lo echara todo a perder.

Primero tenemos una pantalla de presentación en la que se explica la función y el manejo del programa, para, después de ver una pequeña demostración del uso del CRTC, aparecer la pantalla de trabajo con un informe del estado actual de los registros, el valor estándar entre corchetes y el nombre de los mismos.

Abajo, a la derecha, y recuadrado, tenemos los dos Modos de funcionamiento, apareciendo en vídeo inverso el modo activo.

En modo **«Ejecutar»** podemos modificar los registros sin que se observe su efecto, hasta pulsar las teclas CTRL+E.

En modo **«Inmediato»,** nada más modificar un registro, se ejecuta la acción inmediatamente.

Se puede cambiar de modo en cualquier momento pulsando las teclas SHIFT.

En cuanto seleccionamos un registro, su nombre aparece en vídeo inverso y el programa espera la pulsación de una de las teclas posibles que se visualizan en la última línea y que son:

«V»: Visualiza el rango permitido y espera la entrada del dato a enviar al registro.

«Cursor Arriba/Cursor Abajo»: Incrementa/Decrementa el contenido del registro en una unidad.

Cualquier otra tecla, así como un valor fuera de rango, anula la acción, produciendo una señal acústica de error.

En la pantalla se informa con un asterisco del último registro modificado, y, en vídeo inverso aparecerán los valores que no sean estándar, al objeto de una más fácil localización.

Si se llega a perder el control de la pantalla, pulsando la tecla «N» volvemos a la pantalla normal, apareciendo el mensaje: PULSE UNA TECLA PARA VOLVER AL ESTADO ANTERIOR.

Si en ese momento estamos en modo Inmediato y pulsamos cualquier tecla, excepto SHIFT, volveremos al caso anterior; pero si pulsamos SHIFT, cambiamos al modo Ejecutar, por lo que, al pulsar otra tecla, se mantiene la pantalla normal, pudiendo restaurar los registros al valor que queramos hasta que pulsemos las teclas CTRL+E para observar el efecto que se produce.

Con la tecla «X» veremos cuatro de las infinitas posibilidades que nos presenta el manejo del CRTC.

Por fin, pulsando «T» termina el programa restaurando los valores normales del CRTC.

Finalizar aclarando que el registro **r3**, Ancho del Sincronismo, es muy inestable, por lo que el programa no permite acceder al mismo. No obstante, el que quiera aventurarse sólo tiene que modificar la línea 710 quitando la condición **«OR tec=4».** ¡Ah!, y que alterar los registros que se refieren al Cursor no produce efecto alguno.

```
10 '*---------------*
20 '*  CONTROL CRT  *
30 '* por  J.F.Bayo *
40 '*   Mayo 1986   *
50 '*---------------*
60 :
70 '::: PRESENTACION :::
80 :
90 MODE 2:WINDOW 8,80,1,25
100 i$=CHR$(24)
110 PRINT ,i$ SPC(7) "***  CONTROL
DEL CRT  ***" SPC(7)i$:PRINT
120 PRINT"  El programa permite est
udiar los efectos que se producen e
n la"
130 PRINT"pantalla al cambiar el co
ntenido de los registros del CRTC (
Con-"
140 PRINT"trolador del Tubo de Rayo
s Catodicos) utilizando la orden OU
T de"
150 PRINT"BASIC.":PRINT
160 PRINT"  Para modificar un regis
tro pulsamos la tecla correspondien
te a"
170 PRINT"su valor en hexadecimal,
iluminandose el nombre del mismo; y
 el"
180 PRINT"dato se cambia mediante:"
:PRINT
190 PRINT,"V para entrar un dato, s
eguido de ENTER."
200 PRINT, CHR$(240)+" Incrementa e
l dato, y "+CHR$(241)+" Decrementa
el dato."
210 PRINT"  Aparte :"
220 PRINT,"N restaura la pantalla N
ormal."
230 PRINT,"T para Terminar (previa
restauracion)."
240 PRINT,"X Muestra cuatro ejemplo
s":PRINT
250 PRINT"  El programa no permite
entrar datos que puedan bloquear el
 or-"
260 PRINT"denador. La ultima linea
informa de las teclas permitidas.":
PRINT
270 PRINT"  Hay dos Modos de funcio
namiento: Inmediato y  Ejecutar, co
nmu-"
280 PRINT"tables con la tecla (SHIF
T) en cualquier momento.":PRINT
290 PRINT TAB(23) i$+" PULSE UNA TE
CLA "+i$
300 :
310 '::: INICIALIZACION :::
320 :
330 DEFINT a-u
340 DIM stat(3,15),nom$(15)
350 FOR a=0 TO 15
360 READ stat(0,a),stat(1,a),stat(2
,a),nom$(a)
370 stat(3,a)=stat(2,a)
380 f$=MID$(STR$(stat(3,a)),2)
390 f$=SPACE$(3-LEN(f$))+f$
400 nom$(a)="["+f$+"] "+nom$(a)
410 NEXT
420 ENT -1,1,10,1:ENT -2,1,-10,1
430 ENV 1,7,1,1,7,-1,1:ENV 2,1,10,1
,10,1,1,15,5,1,15,-1,15
440 masc1$=i$+"&"+i$:masc2$=i$+" ##
# "+i$
450 m$(0)="Ejecutar ":m$(1)="Inmedi
ato"
460 CALL &BB06:BORDER 12
470 REM MIN,MAX,STANDAR,NOMBRE DEL
REGISTRO
480 DATA 46,100,63,Total horizontal

490 DATA 0,64,40,Horizontal Visuali
zado
500 DATA 0,63,46,Posicion sincronis
mo horizontal
510 DATA 0,255,142,Ancho del sincro
nismo (No Accesible)
520 DATA 0,255,38,Total Vertical
530 DATA 0,255,0,Ajuste total Verti
cal
540 DATA 0,50,25,Vertical visualiza
do
550 DATA 0,255,30,Posicion sincroni
smo vertical
560 DATA 0,255,0,Modo solapado o se
sgado
570 DATA 0,255,7,Numero lineas de b
arrido
580 DATA 0,255,0,Comienzo del curso
r
590 DATA 0,255,0,Fin del cursor
600 DATA 0,255,48,Direccion comienz
o rastreo (byte alto)
610 DATA 0,255,0,Idem (byte bajo)
620 DATA 0,255,192,Registro del cur
sor (byte alto)
630 DATA 0,255,0,Idem (byte bajo)
640 :
650 GOSUB 1400:'*** DEMOSTRACION **
*
660 :
670 '::: CONTROL ENTRADAS :::
680 :
690 GOSUB 840:'inkey$
700 tec=INSTR("0123456789ABCDEFNTX"
+CHR$(5),q$)
710 IF tec=0 OR tec=4 THEN SOUND 7,
600,-2,8,1,2:GOTO 690
720 IF tec=17 THEN GOSUB 1200:GOTO
690
730 IF tec=18 THEN 1200
740 IF tec=19 THEN GOSUB 1340:GOSUB
 1300:GOTO 690
750 IF tec=20 THEN GOSUB 1270:GOTO
690
760 reg=tec-1
770 LOCATE 20,reg+3:PRINT USING mas
c1$;nom$(reg)
780 LOCATE 20,23:PRINT"(V) (Flechas
 cursor) Cualquier otra anula la ac
cion"
790 GOSUB 840
800 IF q$="v" OR q$="V" THEN LOCATE
 1,23:PRINT CHR$(18)+CHR$(11):PRINT
"ENTRAR DATO ("stat(0,reg)"..."stat
(1,reg)") (ENTER para volver):";:LI
NE INPUT "",v$:v=VAL(v$):IF STR$(v)
=" "+v$ THEN IF v<=stat(1,reg) AND
v>=stat(0,reg) THEN stat(3,reg)=v:G
OTO 890
810 IF q$=CHR$(240) AND stat(3,reg)
<stat(1,reg) THEN stat(3,reg)=stat(
3,reg)+1:GOTO 890
820 IF q$=CHR$(241) AND stat(3,reg)
>stat(0,reg) THEN stat(3,reg)=stat(
3,reg)-1:GOTO 890
830 SOUND 7,600,-2,8,1,1:LOCATE 20,
reg+3:PRINT nom$(reg):GOSUB 1300:GO
TO 690
840 IF INKEY(21)<>-1 THEN m=m XOR 1
:LOCATE 48,20+m:PRINT USING masc1$;
```

```
m$(m):LOCATE 48,21-m:PRINT m$(1-m):
GOSUB 1830
850 q$=UPPER$(INKEY$):IF q$="" THEN
 840 ELSE RETURN
860 :
870 '::::: OUT :::::
880 :
890 IF m=0 THEN 910
900 OUT &BC00,reg:OUT &BD00,stat(3,
reg)
910 IF TEST(140,84)<>1 THEN GOSUB 1
130
920 LOCATE 20,reg+3:PRINT nom$(reg)
930 LOCATE 8,reg+3:PRINT"*"
940 LOCATE 15,reg+3:IF stat(3,reg)=
stat(2,reg) THEN PRINT USING " ###
";stat(3,reg) ELSE PRINT USING masc
2$;stat(3,reg)
950 IF reg<>ureg THEN LOCATE 8,ureg
+3:PRINT" "
960 ureg=reg
970 GOSUB 1300:GOTO 490
980 :
990 '::: ESTADO DE LOS REGISTROS ::
:
1000 :
1010 ORIGIN 0,0:DRAW 0,399:DRAW 639
,399:DRAW 639,0:DRAW 0,0:MOVE 1,0:D
RAWR 0,399:MOVE 638,0:DRAWR 0,399
1020 FOR a=0 TO 20:INK 0,12:INK 0,6
:INK 0,24:NEXT:INK 0,1
1030 PRINT TAB(6) i$+" ESTADO ACTUA
L, ESTANDAR Y NOMBRE DE LOS REGISTR
OS DEL CRTC "+i$
1040 PRINT
1050 FOR a=0 TO 15
1060 PRINT TAB(10)"r "+CHR$(a+48-(a
>9)*7)+" =";:PRINT USING " ### ";st
at(3,a):LOCATE 20,a+3:PRINT nom$(a)
1070 BORDER 26-a:INK 1,9+a:NEXT
1080 LOCATE 48,20:PRINT USING masc1
$;m$(0)
1090 LOCATE 48,21:PRINT m$(1)
1100 PLOT 404,84:DRAWR 144,0:DRAWR
0,-40:DRAWR -144,0:DRAWR 0,40
1110 PLOT 482,70:TAG:PRINT"<CTRL+E>
";:TAGOFF
1120 GOTO 1160
1130 LOCATE 5,20:PRINT "* indica el
 ultimo registro modificado"
1140 PRINT TAB(5)"En video inverso
los datos no Estandar"
1150 PLOT 60,84:DRAWR 312,0:DRAWR 0
,-40:DRAWR -312,0:DRAWR 0,40
1160 GOSUB 1300:RETURN
1170 :
1180 '::: PANTALLA NORMAL :::
1190 :
1200 FOR a=0 TO 15:OUT &BC00,a:OUT
&BD00,stat(2,a):NEXT
1210 IF tec=19 THEN RETURN
1220 IF tec=18 THEN LOCATE 1,1:END
1230 LOCATE 1,23:PRINT CHR$(18)
1240 LOCATE 10,23:PRINT i$+" PULSE
UNA TECLA PARA VOLVER AL ESTADO ANT
ERIOR "+i$
1250 GOSUB 840
1260 GOSUB 1300:IF m=0 THEN RETURN
```

```
ESTADO ACTUAL, ESTANDAR Y NOMBRE DE LOS REGISTROS DEL CRTC

r 0 =  63 [ 63] Total horizontal
r 1 =  40 [ 40] Horizontal Visualizado
r 2 =  46 [ 46] Posicion sincronismo horizontal
r 3 = 142 [142] Ancho del sincronismo (No Accesible)
r 4 =  38 [ 38] Total Vertical
r 5 =   0 [  0] Ajuste total Vertical
r 6 =  25 [ 25] Vertical visualizado
r 7 =  30 [ 30] Posicion sincronismo vertical
r 8 =   0 [  0] Modo solapado o sesgado
r 9 =   7 [  7] Numero lineas de barrido
r A =   0 [  0] Comienzo del cursor
r B =   0 [  0] Fin del cursor
r C =  48 [ 48] Direccion comienzo rastreo (byte alto)
r D =   0 [  0] Idem (byte bajo)
r E = 192 [192] Registro del cursor (byte alto)
r F =   0 [  0] Idem (byte bajo)

                                 Ejecutar
                                 Inmediato <CTRL+E>

Teclas posibles: 0,1,2,4,5,6,7,8,9,A,B,C,D,E,F <N> <T> <X>=Ejemplos.
```

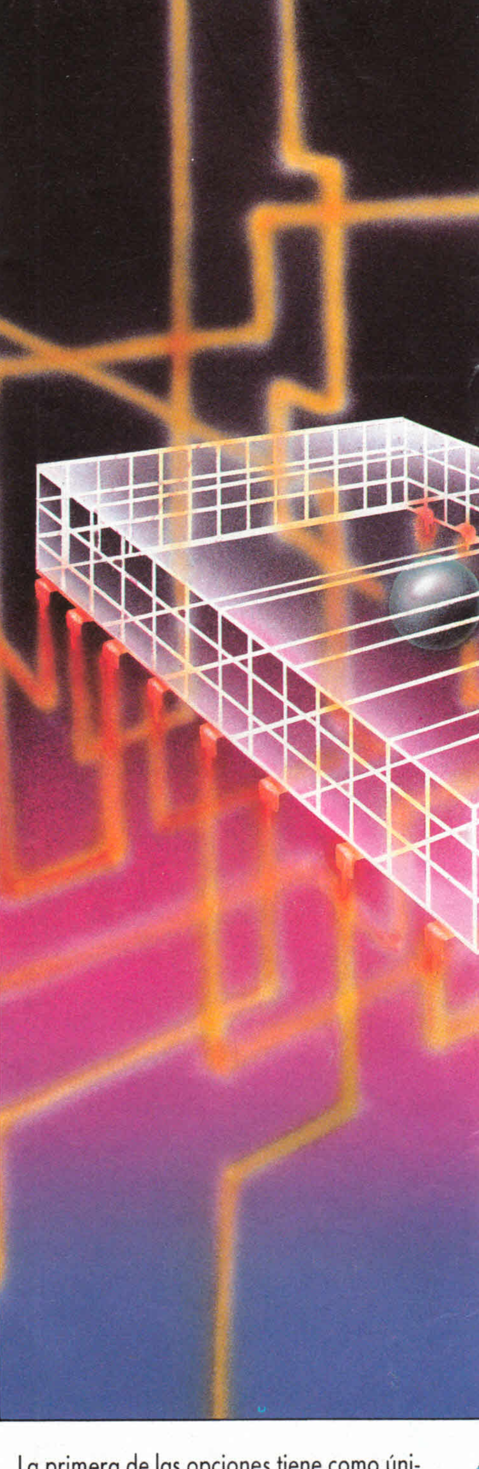
```
1270 FOR a=0 TO 15:OUT &BC00,a:OUT
&BD00,stat(3,a):NEXT
1280 RETURN
1290 :
1300 LOCATE 1,23:PRINT i$+" Teclas
posibles: "+i$+" 0,1,2,4,5,6,7,8,9,
A,B,C,D,E,F <N> <T> <X>=Ejemplos."+
CHR$(18):RETURN
1310 :
1320 ':::: EJEMPLOS ::::
1330 :
1340 GOSUB 1200
1350 LOCATE 1,23:PRINT"ELEGIR EJEMP
LO (1,2,3,4) <ENTER para volver>"+C
HR$(18)
1360 GOSUB 840
1370 ej=INSTR("1234",q$):IF ej=0 TH
EN RETURN
1380 ON ej GOSUB 1540,1620,1700,179
0:RETURN
1390 :
1400 '=== DEMOSTRACION ===
1410 :
1420 FOR a=25 TO 0 STEP-1:GOSUB 147
0:NEXT
1430 MODE 2:WINDOW 5,79,2,24
1440 FOR a=0 TO 25:GOSUB 1470:NEXT
1450 SOUND 7,0,0,0,2,0,31:OUT &BC00
,8:OUT &BD00,1
1460 GOTO 1480
1470 CALL &BD19:OUT &BC00,1:OUT &BD
00,INT(1.6*a):OUT &BC00,6:OUT &BD00
,a:SOUND 2^(a MOD 3),0,7,13,0,0,a+5
:RETURN
1480 INK 1,8:GOSUB 1010:'ESTADO DE
LOS REGISTROS
1490 OUT &BC00,8:OUT &BD00,0
1500 RETURN
1510 :
1520 '=== EJEMPLO 1 ===
1530 :
1540 FOR a=40 TO 0 STEP -1:GOSUB 15
80:NEXT
1550 GOSUB 1830
1560 FOR a=0 TO 40:GOSUB 1580:NEXT
1570 RETURN
1580 CALL &BD19:OUT &BC00,1:OUT &BD
00,a:RETURN
1590 :
1600 '=== EJEMPLO 2 ===
1610 :
1620 FOR a=25 TO 0 STEP -1:GOSUB 16
60:NEXT
1630 OUT &BC00,1:OUT &BD00,0:GOSUB
1830:OUT &BC00,1:OUT &BD00,40
1640 FOR a=0 TO 25:GOSUB 1660:NEXT
1650 RETURN
1660 CALL &BD19:OUT &BC00,6:OUT &BD
00,a:RETURN
1670 :
1680 '=== EJEMPLO 3 ===
1690 :
1700 FOR a=40 TO 0 STEP -1:GOSUB 17
40:NEXT
1710 GOSUB 1830
1720 FOR a=0 TO 40:GOSUB 1740:NEXT
1730 RETURN
1740 CALL &BD19:OUT &BC00,1:OUT &BD
00,a:OUT &BC00,2:OUT &BD00,a+6:RETU
RN
1750 RETURN
1760 :
1770 '=== EJEMPLO 4 ===
1780 :
1790 FOR b=0 TO 30:CALL &BD19:FOR a
=0 TO 5:GOSUB 1810:NEXT:FOR a=4 TO
0 STEP -1:GOSUB 1810:NEXT:NEXT
1800 RETURN
1810 OUT &BC00,5:OUT &BD00,a:RETURN
1820 :
1830 FOR t=0 TO 500:NEXT:RETURN
```



Para que tus dedos no realicen el trabajo duro, M.H. AMSTRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicítanoslo.

MODIFICADOR DE CADENAS

Muchos de nosotros, en alguna ocasión, nos hemos visto obligados a cambiar una cadena de caracteres que se repetía en varias ocasiones en un programa Basic.
Aquellos que hayan vivido esta experiencia, conocerán lo monótona y trabajosa que resulta esta tarea.



A partir de ahora, la modificación de cadenas de un programa Basic, resultará lo más sencillo del mundo si nos ayudamos del programa que aparece listado en este artículo.

Debemos advertir en primer lugar que nuestro programa trabaja únicamente con programas que estén salvados en formato fichero.

Si deseamos trabajar con un programa que no se encuentre en este formato, deberemos hacer lo siguiente:

— Cargar el programa en memoria con:

LOAD''PROGRAMA

— Salvar el programa de la forma siguiente:

SAVE''PROGRAMA'',A

de esta forma nuestra rutina ya podrá trabajar con él, puesto que ahora se encontrará salvado como fichero de caracteres o fichero ASCII.

La rutina encargada de efectuar el trabajo está ubicada a partir de la dirección hexadecimal &A000, y tiene una longitud de 1135 bytes. Por lo tanto no podremos trabajar con programas Basic que ocupen dichas direcciones.

Debemos aclarar que esto no es un gran inconveniente, debido a que apenas ningún programa escrito en Basic llega a ocupar esas direcciones, ya que generalmente no suelen ser demasiado extensos.

De lo dicho anteriormente, se deduce que el programa modificador de cadenas no podrá trabajar con programas Basic que superen los 39 K de longitud.

La solución del peor de los casos

Si en alguna ocasión necesitáramos trabajar con programas de gran extensión, que superarán los 39 K de memoria, no deberemos alarmarnos, puesto que todo tiene solución.

Dicha solución sería partir nuestro programa en dos, es decir, deberíamos cargar el programa en memoria y salvar únicamente la mitad, y a continuación volver a cargarlo en memoria y salvar la otra mitad.

Ahora estaríamos en condiciones de trabajar con cada una de las dos mitades sin problemas de memoria.

Una vez hubiésemos finalizado el trabajo cargaríamos el primer programa y a continuación haríamos un MERGE del segundo programa, con lo cual tendríamos el programa otra vez completo y modificado.

Realizadas estas advertencias y apuntadas las posibles soluciones a los problemas con los que nos podemos encontrar, pasaremos ahora a explicar el manejo del programa modificador de cadenas.

El programa

Con lo primero que nos encontramos al ejecutar el programa, es con un menú de ayuda en el que se encuentran las siguientes opciones:

ORDENES DEL MODIFICADOR
B — Basic
L — Listar
C — Cargar (LOAD)
S — Salvar (SAVE)
F — Cambiar cadenas

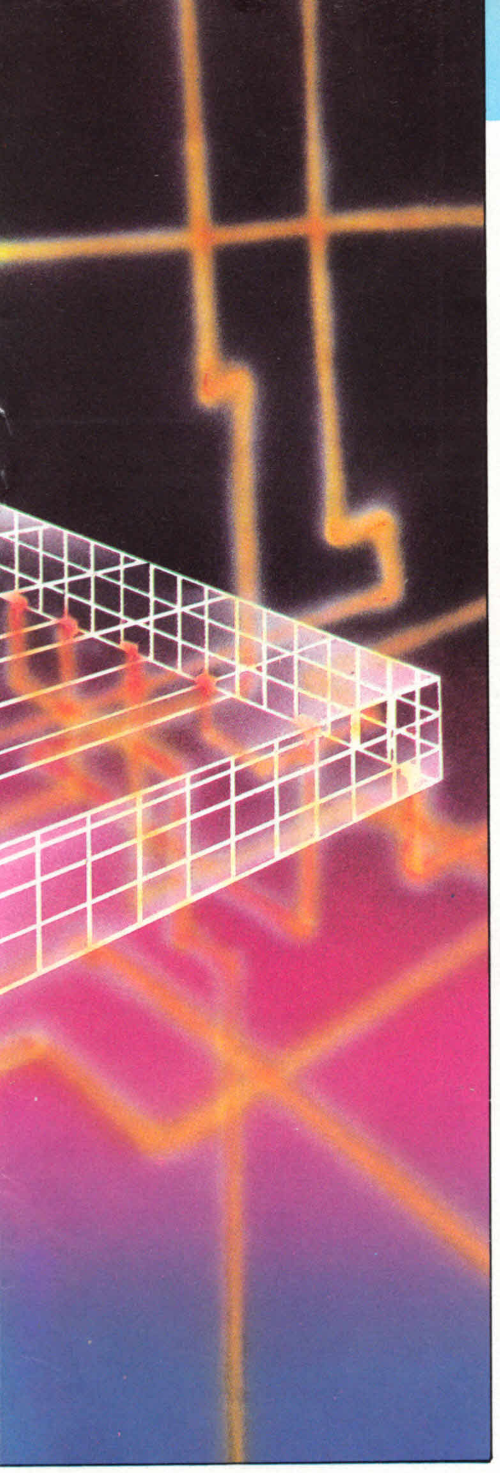
La primera de las opciones tiene como único objeto, permitirnos retornar al Basic.

La opción **''LISTAR''**, nos permitirá listar el programa que se encuentre en memoria en ese momento. Si no hemos cargado ningún programa, al elegir esta opción, aparece un mensaje indicándonos que no existen ningún programa en memoria.

Cuando se elija la opción **CARGAR**, se nos pedirá el nombre del programa que deseamos cargar en memoria. Una vez escrito dicho nombre, deberemos pulsar **''ENTER''**.

Mientras se esté ejecutando la carga del programa, aparecerán unas líneas en la pantalla, pero eso no nos debe asustar, ya que dichas rayas son debidas a que se ha elegido dicha zona de memoria como buffer de carga.

Se ha hecho de esta forma con el fin de ahorrar memoria ya que dicho buffer necesita: 2 k.



A continuación se nos preguntará cuál es la cadena que debe sustituir a la anterior.

Cuando se hayan dado estos parámetros, el programa buscará una cadena idéntica a la primera que hayamos dado, y la cambiará por la última.

Cuando haya finalizado el trabajo, se imprimirá un mensaje indicando que dichas cadenas se han modificado. En el caso de que no encuentre ninguna cadena de caracteres igual a la que hemos introducido, nos lo indicará con otro mensaje.

Funcionamiento y rutinas principales

Vamos a ver ahora cuáles son las rutinas más importantes de que se compone nuestro programa.

Para hacernos una idea de conjunto del programa realizaremos una especie de diagrama de flujo con el cual podremos ver globalmente los pasos a seguir.

lectura del teclado. La rutina permanecerá en dicho bucle hasta que no sea pulsada alguna de las teclas predeterminadas.

Cuando se detecte la pulsación de alguna de las teclas que indican al programa que se ha elegido una opción, se enviará el control a la rutina que corresponda, y una vez finalizado el trabajo se retornará al bucle principal de lectura de teclado.

La primera subrutina con la que nos encontramos en el programa, es la encargada de efectuar la carga.

Para averiguar cuál es el nombre del programa que se desea cargar, efectuamos una llamada a la rutina INPUT, la cual nos pedirá el nombre del fichero que deseamos introducir en la memoria del ordenador.

Hecho esto se retorna a la rutina LOAD para efectuar dicha carga, utilizando las rutinas del firmware capaces de leer un programa en forma de fichero ASCII.

Una vez finalizada su tarea, reinicializa la pantalla y devuelve el control del programa al bucle principal para detectar la elección de otras opciones.

DIAGRAMA DE FLUJO DEL MODIFICADOR

Otra de las opciones que posee es la que permite salvar el programa que tenemos en memoria en disco o cinta. En esta opción volverán a aparecer las rayas mencionadas anteriormente, debido a que se utiliza el mismo buffer de memoria.

Por último, se nos ofrece la posibilidad de modificación de cadenas de caracteres. Esta opción nos permitirá cambiar una cadena por otra de una longitud no superior a 20 caracteres. Si la longitud de la cadena de caracteres que se desea modificar es superior a dicha cifra, entonces se deberá efectuar el cambio en dos pasadas, es decir, deberemos modificar en primer lugar los primeros 20 caracteres y a continuación los restantes. Cuando deseemos modificar alguna cadena, lo primero que hará el programa será preguntarnos cuál es la cadena que se desea cambiar; una vez introducida, deberemos pulsar "ENTER".

En primer lugar hacemos una inicialización de todos los parámetros que utilizaremos durante la ejecución del programa, seguidamente preparamos la pantalla eligiendo el modo en el que vamos a trabajar.

Activamos una ventana, en la cual imprimiremos el menú de opciones. En este caso se ha elegido una situada en la parte superior de la pantalla.

Una vez hecho esto, entramos en el bucle principal del programa, donde se produce la

Rutina de listar

Otra de las rutinas que componen el programa es la encargada de producir un listado en pantalla. Para ello se toma la dirección inicial del programa, y a partir de ahí va tomando los caracteres que encuentra en memoria y los imprime en pantalla.

Cuando se detecta que se ha llegado al final del programa, retorna al bucle principal.

A continuación nos encontramos con la rutina más importante de nuestro programa, se trata de la rutina encargada de modificar las cadenas de caracteres.

En primer lugar, se llama a la rutina INPUT, para conseguir la información de qué cadena debe modificar y por cuál ha de sustituirla. Cuando ha conseguido dichos datos, intenta detectar una cadena idéntica a la dada, sino lo consigue devuelve el control al bucle principal imprimiendo el mensaje de que no ha encontrado ninguna cadena.

En caso de que se encuentre con una cadena de caracteres idéntica a la que se ha dado, entonces se encarga de sustituirla por la nueva, reservando el espacio de memoria necesario para la nueva cadena de caracteres a introducir, y eliminando al mismo tiempo el espacio de memoria que ocupaba la antigua.

Una vez realizada la misma operación con todas las cadenas de caracteres idénticas a la dada, imprime el mensaje correspondiente y retorna al bucle de lectura de teclado.

Por último nos queda comentar la rutina de SAVE, ésta hace una llamada a INPUT para saber qué nombre debe dar al nuevo fichero, y a continuación se almacena en cinta o disco.

Después de finalizar su trabajo, inicializa la pantalla y retorna al bucle principal.

Una vez realizada la descripción del funcionamiento de la rutina y dadas las instrucciones para su correcto funcionamiento, únicamente queda copiar el listado de dicho programa.

Manipulación del listado

Para ello, deberemos copiar el listado ensamblador que aparece al final del artículo, y salvarlo en cinta o disco.

También ofrecemos un cargador Basic, para aquellos que prefieran teclearlo directamente en forma de DATAS.

Aquellos que elijan esta última opción, deberán ejecutar el programa cargador una vez tecleado, y en el caso de que aparezca algún mensaje de error, se deberán revisar las líneas DATA.

Si dicho mensaje no aparece, indicará que todo ha ido bien, y por lo tanto estaremos en condiciones de salvarlo en cinta o disco.

Para ello haremos lo siguiente:

```
SAVE"CADENAS",B,&A000,1135
```

Cuando se quiera trabajar con el modificador de cadenas, deberemos escribir un programa BASIC como el que se indica a continuación:

```
10 MEMORY &9FF
20 LOAD"CADENAS)),&A000
30 CALL &A000
```

No queda nada más que decir, sólo desear que este programa os sea de gran utilidad.

```
;MODIFICADOR DE CADENAS
        ORG  #A000
        LD   HL,0
        LD   (FINPRO),HL
        CALL VENTAN
        JP   TEC
VENTAN: LD   A,2
        CALL #BC0E
        LD   A,1
        CALL #BBB4
        LD   HL,0
        LD   DE,#5001
        CALL #BB66
        LD   A,1
        CALL #BB96
        XOR  A
        CALL #BB90
        CALL #BB6C
        LD   HL,TXT
OTRPO:  LD   A,(HL)
        CP   255
        JR   Z,FUERA
        CALL #BB5A
        INC  HL
        JR   OTRPO
FUERA:  XOR  A
        CALL #BBB4
        LD   HL,#0002
        LD   DE,#5019
        CALL #BB66
        RET
;
;
TEC:    LD   A,54
        CALL #BB1E
        RET  NZ
        LD   A,36
        CALL #BB1E
        JR   Z,PAST1
        CALL IMPRE
PAST1:  LD   A,60
        CALL #BB1E
        JR   Z,PAST2
        CALL SAVE
PAST2:  LD   A,62
        CALL #BB1E
        JR   Z,PAST3
        CALL LOAD
PAST3:  LD   A,53
        CALL #BB1E
        JR   Z,TEC
        CALL BUSCA
        JR   TEC
;
;
LOAD:   CALL #BB6C
        CALL I_LOAD
        CALL #BB6C
        LD   A,(L_LOAD)
        LD   B,A
        LD   HL,N_LOAD
        LD   DE,#C0A0
        CALL #BC77
        JP   NC,ERROR
        LD   HL,5000
BULEC:  CALL #BC80
        JR   NC,FINIT
        LD   (HL),A
        INC  HL
        JR   BULEC
FINIT:  DEC  HL
        LD   (FINPRO),HL
        CALL #BC7A
        CALL VENTAN
        LD   HL,#080D
        LD   DE,TXTPL
        CALL PRINT
        RET
IMPRE:  CALL #BB6C
        LD   HL,(FINPRO)
        LD   A,H
        OR   L
        JP   Z,NOPRO
        LD   DE,5000
OTRA:   PUSH DE
        LD   A,66
        CALL #BB1E
        CALL NZ,PAUSA
        POP  DE
        LD   A,(DE)
        CALL #BB5A
        INC  DE
        LD   HL,(FINPRO)
        LD   A,H
        CP   D
        JR   NZ,OTRA
        LD   A,L
        CP   E
        JR   NZ,OTRA
        RET
PAUSA:  LD   BC,40000
PAUS:   DEC  BC
        LD   A,B
        OR   C
        JR   NZ,PAUS
        CALL #BB03
        CALL #BB18
        RET
;
;BUSCA CADENA
;
BUSCA:  LD   HL,(FINPRO)
        LD   A,H
        OR   L
        JP   Z,NOPRO
        XOR  A
        LD   (NUMCA),A
        CALL #BB6C
        CALL NOMCA1
        CALL #BB6C
        CALL NOMCAN
        CALL #BB6C
        LD   HL,CADENV
        LD   DE,5000
        LD   IX,1000
VULV:   LD   B,0
BUC:    LD   A,(LONCV)
        CP   B
        JR   Z,FIN
        PUSH HL
        LD   HL,(FINPRO)
        LD   A,D
        CP   H
        JR   NZ,SIGU
        LD   A,E
        CP   L
        JR   Z,FINRU
SIGU:   POP  HL
        LD   A,(DE)
        LD   (IX+0),A
        CP   (HL)
        INC  DE
        INC  IX
        JR   Z,INCCO
        LD   B,0
        LD   HL,CADENV
        JR   BUC
INCCO:  INC  B
        INC  HL
        JR   BUC
FIN:    LD   (DIREC),DE
OTRDE:  DEC  IX
        DJNZ OTRDE
        LD   A,(NUMCA)
        INC  A
        LD   (NUMCA),A
        LD   A,(LONCN)
        LD   B,A
        LD   HL,CADENN
PONBU:  LD   A,(HL)
        LD   (IX+0),A
        INC  IX
        INC  HL
        DJNZ PONBU
        LD   HL,CADENV
        JR   VULV
FINRU:  POP  HL
        PUSH IX
        POP  HL
        LD   DE,4000
        ADD  HL,DE
        LD   (FINPRO),HL
        LD   DE,5000
        SCF
        CCF
        SBC  HL,DE
        LD   B,H
        LD   C,L
        LD   HL,1000
        LD   DE,5000
        LDIR
        LD   A,(NUMCA)
        CP   0
        JR   NZ,SIGCA
        LD   HL,#080D
        LD   DE,TXTNOC
        CALL PRINT
        RET
SIGCA:  LD   HL,#080D
        LD   DE,TXTSIC
        CALL PRINT
        RET
SAVE:   CALL #BB6C
        LD   HL,(FINPRO)
        LD   A,H
        OR   L
        JP   Z,NOPRO
        CALL I_SAVE
        CALL #BB6C
        LD   A,(L_SAVE)
        LD   B,A
        LD   HL,N_SAVE
        LD   DE,#C0A0
        CALL #BC8C
        LD   DE,5000
OTRCA:  LD   A,(DE)
        CALL #BC95
        LD   HL,(FINPRO)
        LD   A,H
        CP   D
        JR   NZ,PASI
        LD   A,L
        CP   E
        JR   Z,FINSA
PASI:   INC  DE
        JR   OTRCA
FINSA:  CALL #BC8F
        CALL #BC92
        CALL VENTAN
        RET
NOMCA1: LD   A,20
        LD   (MIRLON+1),A
        LD   HL,#080D
        LD   DE,TXT0
        CALL PRINT
        CALL INPUT
        LD   A,(LONG)
        LD   (LONCV),A
        LD   DE,CADENV
        CALL TRASP
        RET
NOMCAN: LD   A,20
        LD   (MIRLON+1),A
        LD   HL,#080D
        LD   DE,TXT1
        CALL PRINT
        CALL INPUT
        LD   A,(LONG)
        LD   (LONCN),A
        LD   DE,CADENN
        CALL TRASP
        RET
I_LOAD: LD   A,8
        LD   (MIRLON+1),A
        LD   HL,#080D
        LD   DE,TXT2
        CALL PRINT
        CALL INPUT
        LD   A,(LONG)
        LD   (L_LOAD),A
        LD   DE,N_LOAD
        CALL TRASP
        RET
I_SAVE: LD   A,8
        LD   (MIRLON+1),A
        LD   HL,#080D
```

```
2570          LD   DE,TXT3
2580          CALL PRINT
2590          CALL INPUT
2600          LD   A,(LONG)
2610          LD   (L_SAVE),A
2620          LD   DE,N_SAVE
2630          CALL TRASP
2640          RET
2650 INPUT:   CALL #BB03
2660          XOR  A
2670          LD   (LONG),A
2680          LD   DE,NAME
2690          LD   HL,#090F
2700          CALL #BB75
2710          LD   A,"_"
2720          CALL #BB5A
2730 I_BUC:   CALL #BB18
2740          LD   B,A
2750          CP   13
2760          RET  Z
2770          CP   127
2780          JR   Z,DELET
2790          CP   32
2800          JR   C,I_BUC
2810          CP   129
2820          JR   NC,I_BUC
2830          LD   A,(LONG)
2840 MIRLON:  CP   4
2850          JR   Z,I_BUC
2860          LD   A,8
2870          CALL #BB5A
2880          LD   A,B
2890          CALL #BB5A
2900          LD   A,"_"
2910          CALL #BB5A
2920          LD   A,(LONG)
2930          INC  A
2940          LD   (LONG),A
2950          LD   A,B
2960          LD   (DE),A
2970          INC  DE
2980          JR   I_BUC
2990 DELET:   LD   A,(LONG)
3000          CP   0
3010          JR   Z,I_BUC
3020          DEC  A
3030          LD   (LONG),A
3040          DEC  DE
3050          LD   A,32
3060          LD   (DE),A
3070          LD   A,8
3080          CALL #BB5A
3090          LD   A,32
3100          CALL #BB5A
3110          LD   A,8
3120          CALL #BB5A
3130          LD   A,8
3140          CALL #BB5A
3150          LD   A,32
3160          CALL #BB5A
3170          LD   A,8
3180          CALL #BB5A
3190          LD   A,"_"
3200          CALL #BB5A
3210          JR   I_BUC
3220 TRASP:   LD   A,(LONG)
3230          LD   C,A
3240          LD   B,0
3250          LD   HL,NAME
3260          LDIR
3270          RET
3280 PRINT:   CALL #BB75
3290 PBUD:    LD   A,(DE)
3300          CP   255
3310          RET  Z
3320          CALL #BB5A
3330          INC  DE
3340          JR   PBUD
3350          RET
3360 NOPRO:   LD   HL,#080D
3370          LD   DE,TXTPR
3380          CALL PRINT
3390          RET
3400 ERROR:   LD   HL,#080D
3410          LD   DE,TXTER
3420          CALL PRINT
3430          RET
3440 TXTO:    DEFM "NOMBRE DE LA CADENA A CAMBIAR"
3450          DEFB 255
3460 TXT1:    DEFM "NOMBRE DE LA NUEVA CADENA"
3470          DEFB 255
3480 TXT2:    DEFM "NOMBRE DEL PROGRAMA A CARGAR"
3490          DEFB 255
3500 TXT3:    DEFM "NOMBRE DEL PROGRAMA A SALVAR"
3510          DEFB 255
3520 N_LOAD:  DEFS 10
3530 L_LOAD:  DEFS 1
3540 N_SAVE:  DEFS 10
3550 L_SAVE:  DEFS 1
3560 CADENV:  DEFS 20
3570 LONCV:   DEFS 1
3580 CADENN:  DEFS 20
3590 LONCN:   DEFS 1
3600 TXTPR:   DEFM "NO HAY PROGRAMA EN MEMORIA"
3610          DEFB 255
3620 TXTPL:   DEFM "PROGRAMA EN MEMORIA"
3630          DEFB 255
3640 TXTNOC:  DEFM "NO EXISTE ESTA CADENA"
3650          DEFB 255
3660 TXTSIC:  DEFM "CADENAS CAMBIADAS"
3670          DEFB 255
3680 TXTER:   DEFM "ERROR"
3690          DEFB 255
3700 TXT:     DEFM "     B - BASIC    L - LISTAR
 C - CARGAR
3710          DEFM "S - SALVAR   F - CAMBIAR CADENAS"
3720          DEFB 255
3730 FINPRO:  DEFW 0
3740 NUMCA:   DEFB 0
3750 DIREC:   DEFS 2
3760 LONG:    DEFS 1
3770 NAME:    DEFS 25
00

LEC   A093  BUSCA  A0EB
DENV  A37A  DELET  A290
ROR   A2EB  FIN    A137
NPRO  A44C  FINRU  A15B
```

```
10 REM MODIFICADOR DE CADENAS
20 REM PROGRAMA CARGADOR
30 FOR N=&A000 TO &A46B
40 READ A:SUMA=SUMA+A
50 POKE N,A
60 NEXT
70 IF SUMA<>109212 THEN PRINT "ERRO
R EN DATAS"
80 DATA 33,0,0,34,76,164,205
90 DATA 12,160,195,71,160,62,2
100 DATA 205,14,188,62,1,205,180
110 DATA 187,33,0,0,17,1,80
120 DATA 205,102,187,62,1,205,150
130 DATA 187,175,205,144,187,205,10
8
140 DATA 187,33,1,164,126,254,255
150 DATA 40,6,205,90,187,35,24
160 DATA 245,175,205,180,187,33,2
170 DATA 0,17,25,80,205,102,187
180 DATA 201,62,54,205,30,187,192
190 DATA 62,36,205,30,187,40,3
200 DATA 205,176,160,62,60,205,30
210 DATA 187,40,3,205,146,161,62
220 DATA 62,205,30,187,40,3,205
230 DATA 119,160,62,53,205,30,187
240 DATA 40,213,205,232,160,24,208
250 DATA 205,108,187,205,11,162,205
260 DATA 108,187,58,110,163,71,33
270 DATA 100,163,17,160,192,205,119
280 DATA 188,210,232,162,33,136,19
290 DATA 205,128,188,48,4,119,35
300 DATA 24,247,43,34,76,164,205
310 DATA 122,188,205,12,160,33,13
320 DATA 8,17,191,163,205,208,162
330 DATA 201,205,108,187,42,76,164
340 DATA 124,181,202,222,162,17,136
350 DATA 19,213,62,66,205,30,187
360 DATA 196,217,160,209,26,205,90
370 DATA 187,19,42,76,164,124,186
380 DATA 32,234,125,187,32,230,201
390 DATA 1,64,156,11,120,177,32
400 DATA 251,205,3,187,205,24,187
410 DATA 201,42,76,164,124,181,202
420 DATA 222,162,175,50,78,164,205
430 DATA 108,187,205,207,161,205,10
8
440 DATA 187,205,237,161,205,108,18
7
450 DATA 33,122,163,17,136,19,221
460 DATA 33,232,3,6,0,58,142
470 DATA 163,184,40,34,229,42,76
480 DATA 164,122,188,32,4,123,189
490 DATA 40,58,225,26,221,119,0
500 DATA 190,19,221,35,40,7,6
510 DATA 0,33,122,163,24,220,4
520 DATA 35,24,216,237,83,79,164
530 DATA 221,43,16,252,58,78,164
540 DATA 60,50,78,164,58,163,163
550 DATA 71,33,143,163,126,221,119
560 DATA 0,221,35,35,16,247,33
570 DATA 122,163,24,178,225,221,229
580 DATA 225,17,160,15,25,34,76
590 DATA 164,17,136,19,55,63,237
600 DATA 82,68,77,33,232,3,17
610 DATA 136,19,237,176,58,78,164
620 DATA 254,0,32,10,33,13,8
630 DATA 17,211,163,205,208,162,201
640 DATA 33,13,8,17,233,163,205
650 DATA 208,162,201,205,108,187,42
660 DATA 76,164,124,181,202,222,162
670 DATA 205,41,162,205,108,187,58
680 DATA 121,163,71,33,111,163,17
690 DATA 160,192,205,140,188,17,136
700 DATA 19,26,205,149,188,42,76
710 DATA 164,124,186,32,4,125,187
720 DATA 40,3,19,24,238,205,143
730 DATA 188,205,146,188,205,12,160
740 DATA 201,62,20,50,115,162,33
750 DATA 13,8,17,242,162,205,208
760 DATA 162,205,71,162,58,81,164
770 DATA 50,142,163,17,122,163,205
780 DATA 196,162,201,62,20,50,115
790 DATA 162,33,13,8,17,16,163
800 DATA 205,208,162,205,71,162,58
810 DATA 81,164,50,163,163,17,143
820 DATA 163,205,196,162,201,62,8
830 DATA 50,115,162,33,13,8,17
840 DATA 42,163,205,208,162,205,71
850 DATA 162,58,81,164,50,110,163
860 DATA 17,100,163,205,196,162,201
870 DATA 62,8,50,115,162,33,13
880 DATA 8,17,71,163,205,208,162
890 DATA 205,71,162,58,81,164,50
900 DATA 121,163,17,111,163,205,196
910 DATA 162,201,205,3,187,175,50
920 DATA 81,164,17,82,164,33,15
930 DATA 9,205,117,187,62,95,205
940 DATA 90,187,205,24,187,71,254
950 DATA 13,200,254,127,40,41,254
960 DATA 32,56,241,254,129,48,237
970 DATA 58,81,164,254,4,40,230
980 DATA 62,8,205,90,187,120,205
990 DATA 90,187,62,95,205,90,187
1000 DATA 58,81,164,60,50,81,164
1010 DATA 120,18,19,24,204,58,81
1020 DATA 164,254,0,40,197,61,50
1030 DATA 81,164,27,62,32,18,62
1040 DATA 8,205,90,187,62,32,205
1050 DATA 90,187,62,8,205,90,187
1060 DATA 62,8,205,90,187,62,32
1070 DATA 205,90,187,62,8,205,90
1080 DATA 187,62,95,205,90,187,24
1090 DATA 152,58,81,164,79,6,0
1100 DATA 33,82,164,237,176,201,205
1110 DATA 117,187,26,254,255,200,20
5
1120 DATA 90,187,19,24,246,201,33
1130 DATA 13,8,17,164,163,205,208
1140 DATA 162,201,33,13,8,17,251
1150 DATA 163,205,208,162,201,78,79
1160 DATA 77,66,82,69,32,68,69
1170 DATA 32,76,65,32,67,65,68
1180 DATA 69,78,65,32,65,32,67
1190 DATA 65,77,66,73,65,82,255
1200 DATA 78,79,77,66,82,69,32
1210 DATA 68,69,32,76,65,32,78
1220 DATA 85,69,86,65,32,67,65
1230 DATA 68,69,78,65,255,78,79
1240 DATA 77,66,82,69,32,68,69
1250 DATA 76,32,80,82,79,71,82
1260 DATA 65,77,65,32,65,32,67
1270 DATA 65,82,71,65,82,255,78
1280 DATA 79,77,66,82,69,32,68
1290 DATA 69,76,32,80,82,79,71
1300 DATA 82,65,77,65,32,65,32
1310 DATA 83,65,76,86,65,82,255
1320 DATA 0,0,0,0,0,0,0
1330 DATA 0,0,0,0,0,0,0
1340 DATA 0,0,0,0,0,0,0
1350 DATA 0,0,0,0,0,0,0
1360 DATA 0,0,0,0,0,0,0
1370 DATA 0,0,0,0,0,0,0
1380 DATA 0,0,0,0,0,0,0
1390 DATA 0,0,0,0,0,0,0
1400 DATA 0,0,0,0,0,0,0
1410 DATA 0,78,79,32,72,65,89
1420 DATA 32,80,82,79,71,82,65
1430 DATA 77,65,32,69,78,32,77
1440 DATA 69,77,79,82,73,65,255
1450 DATA 80,82,79,71,82,65,77
1460 DATA 65,32,69,78,32,77,69
1470 DATA 77,79,82,73,65,255,78
1480 DATA 79,32,69,88,73,83,84
1490 DATA 69,32,69,83,84,65,32
1500 DATA 67,65,68,69,78,65,255
1510 DATA 67,65,68,69,78,65,83
1520 DATA 32,67,65,77,66,73,65
1530 DATA 68,65,83,255,69,82,82
1540 DATA 79,82,255,32,32,32,32
1550 DATA 66,32,45,32,66,65,83
1560 DATA 73,67,32,32,32,76,32
1570 DATA 45,32,76,73,83,84,65
1580 DATA 82,32,32,32,67,32,45
1590 DATA 32,67,65,82,71,65,82
1600 DATA 32,32,32,83,32,45,32
1610 DATA 83,65,76,86,65,82,32
1620 DATA 32,32,70,32,45,32,67
1630 DATA 65,77,66,73,65,82,32
1640 DATA 67,65,68,69,78,65,83
1650 DATA 255,0,0,0,0,0,0
1660 DATA 0,0,0,0,0,0,0
1670 DATA 0,0,0,0,0,0,0
1680 DATA 0,0,0,0,0,0,0
1690 DATA 0,0,0,0,0,0,0
```



Para que tus dedos no realicen el trabajo duro, M.H. AMSTRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicítanoslo.

LISTADOR DE VARIABLES

Otro de los programas de utilidades que os ofrecemos en este número, es el listador de variables, utilizable a través de nuevos comandos RSX.

Mediante dichos comandos, estaremos en condiciones de listar tanto las variables numéricas como alfanuméricas que se utilicen en un programa.

La rutina en código máquina que realiza dicha función, está ubicada a partir de la dirección hexadecimal $A000 y tiene una longitud de 585 bytes.

Debido a la zona de memoria que ocupa nuestro programa, no podremos trabajar con programas de gran longitud, ya que de lo contrario corromperíamos esas direcciones de memoria. Así pues, la longitud de nuestros programas, no debe superar 39k.

Para que dicha rutina funcione correctamente, deberemos ejecutar el siguiente programa Basic:

```
10 MEMORY &9FFF
20 LOAD"RSX
30 CALL &A000
```

Una vez hecho esto, borraremos todo lo que hay en la memoria mediante el comando:

NEW

y a continuación podremos cargar en memoria el programa con el cual vayamos a trabajar.

La línea 30 del anterior programa Basic, se utiliza para indicar al ordenador que existen nuevos comandos, ya que de lo contario, los ignoraría.

Una vez tengamos en memoria el programa con el que deseemos trabajar, **no deberemos ejecutarlo,** ya que de lo contrario se inicializarían las variables, y la rutina sería incapaz de identificarlas.

A partir de este momento, estamos en condiciones de utilizar nuestros nuevos comandos.

Vamos a indicar a continuación cuáles son dichos comandos y de qué forma funcionan.

El primero de ellos se utiliza para listar las variables alfanuméricas, y se puede usar de dos formas distintas, según la opción deseada.

Este nuevo comando es el siguiente:

IVALFA

escrito de esta forma, dicho comando produciría un listado de todas las variables alfanuméricas, indicando asimismo la línea en la que cada una de ellas se encuentra.

Si por ejemplo existiera en memoria un programa que tuviera la variable alfanumérica A$ en las líeas 20, 40, la variable HOLA$ en las líneas 50 y 100 y la variable B$ en las líneas 20, 50 y 60, tras la ejecución del anterior comando, nos aparecería el siguiente listado:

```
00020 A$ B$
00040 A$
00050 B$ HOLA$
00060 B$
00100 HOLA$
```

Otra manera de utilizarlo sería de la forma que se indica a continuación:

IVALFA, **"HOLA"**

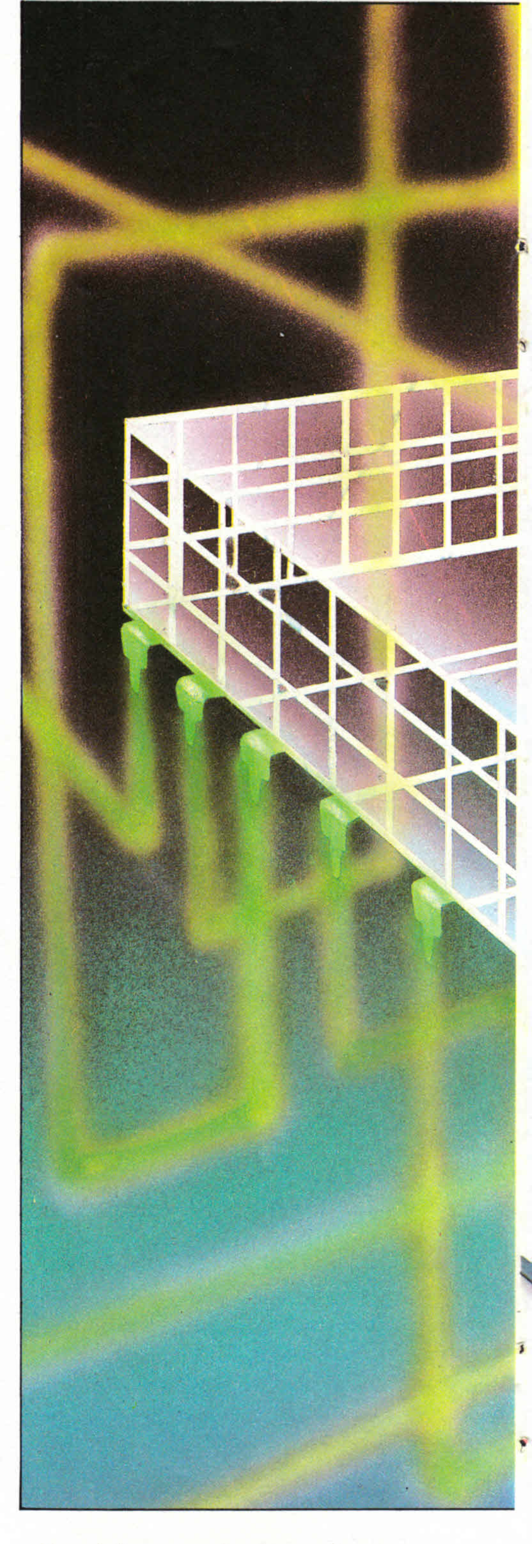
La ejecución de este comando tal como está escrito, produciría el listado de la variable alfanumérica 'HOLA$', indicando al mismo tiempo en qué líneas de programa se encuentra.

Utilizando el ejemplo del hipotético programa citado anteriormente, este comando, nos produciría un listado en pantalla, como el siguiente:

```
00050 HOLA$
00100 HOLA$
```

Este último comando nos será de utilidad cuando se desee conocer en qué partes del programa actúa una variable alfanumérica, y en particular en qué líneas de programa se está utilizando.

Así pues, cuando se ejecute, se producirá la búsqueda de la variable introducida, y en el caso de ser encontrada se imprimirá en pantalla. En el caso de que no exista ninguna variable de este tipo, no se producirá ningún tipo de impresión en pantalla.

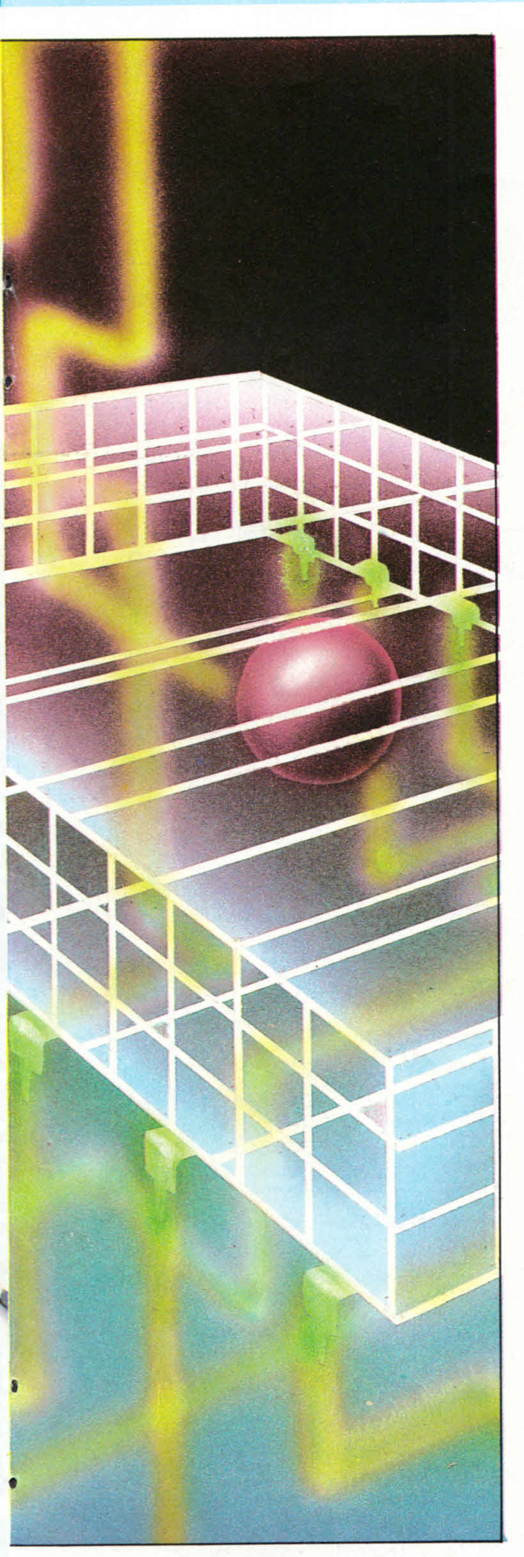


Otro de los comandos de que disponemos, es el que se indica a continuación:

IVNUME

Dicho comando provoca el listado de todas las variables numéricas que contenga nuestro programa, así como los números de línea donde se encuentran.

Como hemos hecho anteriormente, supondremos que existe en memoria un programa que contiene en las líneas 10 y 60 la variable numérica 'POSX', en las líneas 20 y 90 la variable 'YPOS', y en la línea 40 la variable 'DI-



NERO'. La ejecución del anterior comando, produciría un listado semejante al que se indica:

00010 POSX
00020 YPOS
00040 DINERO
00060 POSX
00090 YPOS

La otra variante de este último comando, nos permitirá buscar una variable numérica en particular, produciéndose únicamente la impresión de dicha variable en pantalla, así como los números de línea en que se encuentra.

Este comando es el siguiente:

IVNUME, **"POSX"**

Su ejecución provoca la busqueda de la cadena introducida entre comillas. En el caso de tener en memoria el programa anteriormente mencionado, provocaría la impresión en pantalla del siguiente texto:

00010 POSX
00060 POSX

También por impresora

El último comando del que disponemos, y no por ello menos importante es:

IP

el cual nos ofrece la opción de imprimir en pantalla o bien en impresora.

Este comando actúa como una especie de commutador, es decir cuando, se pulsa, activa lo que anteriormente estuviera desactivado.

Así, por ejemplo, si estamos produciendo un listado de variables en pantalla y ejecutando dicho comando, obligaríamos a que el listado saliera por impresora. Si ahora deseamos volver a listar en pantalla, únicamente deberemos ejecutarlo otra vez.

Vamos a ver a continuación cuáles son las principales rutinas de que consta nuestro programa.

En primer lugar se efectúa la instalación de los nuevos comandos RSX en el sistema; para ello se definen dichos comandos y se efectúa una llamada al firmware, que anuncia al sistema su creación.

La primera rutina con que nos encontramos es la que se encarga de activar el comando con parámeteros, o bien sin utilizar ningún parámetro.

Registro A=0 indica que no hay parámetros
Registro A < > 0 indica la existencia de parámetros.

De esta forma, el programa distinguiría las posibles opciones del comando RSX.

Cuando no se utilicen parámetros, el programa saltará inmediatamente a la rutina encargada de buscar todas las variables alfanuméricas que existan en el programa.

Para detectar dichas variables, se ha definido en un buffer la cadena de bytes por la que pueden ser identificadas, y que es la siguiente:

Variable alfanumérica... 3,0,0,nombre

dado que en este caso el nombre de la variable no nos interesa, únicamente se buscará una cadena que contenga los tres primeros bytes.

Una vez encontrada, conoceremos la existencia de una variable alfanumérica en esa dirección de memoria. Para reconocer de qué variable se trata tomaremos los bytes que siguen a continuación, hasta encontrarnos con uno que contenga el valor de un carácter ASCII al que se le ha sumado 128, ya que es de esta forma como el Basic almacena dichas variables.

Veamos por ejemplo cómo se encontraría almacenada en memoria la variable HOLA$:

3,0,0,**"H", "O", "L", "A"**+128

Cuando se utilice el anterior comando con parámetros, en primer lugar se reconocerá qué parámetros se han introducido, y los colocará en un buffer, detrás de los bytes 3,0,0 identificativos de variable alfanumérica.

A continuación se enviará el control del programa a la rutina encargada de buscar cadenas en memorias. Así por ejemplo si el parámetro es la cadena "POSX", dicha rutina intentará localizar la siguiente secuencia de números:

3,0,0,**"P", "O", "S", "X"**+128

Por último nos queda por describir el comando encargado de localizar las variables numéricas. La rutina utilizada, también chequea en primer lugar el contenido del acumulador para averiguar si se han enviado o no parámetros.

En el caso de que no existan dichos parámetros, se llama a la rutina encargada de localizar este tipo de variables, las cuales pueden ser reconocidas por la siguiente secuencia de bytes:

13,0,0

De esta forma, cuando se encuentre la cadena, se conocerá la existencia de una variable numérica en esa dirección de memoria, por lo que tomaremos los bytes que siguen hasta encontrarnos con uno que contenga el valor de un carácter ASCII más 128, para conocer cuál es el nombre de dicha variable.

Como podemos comprobar este tipo de variables se almacenan en memoria igual que las tratadas anteriormente.

Cuando se utilice este último comando con algún parámetro, se tomará dicho parámetro y se colocará en un buffer a continuación de los bytes indicadores de variable numérica, y se llamará a la rutina de búsqueda para que los localice, y una vez encontrados se imprimirán en pantalla.

Una vez dadas las instrucciones de funcionamiento y explicadas las rutinas más interesantes, estamos en condiciones de poder utilizar correctamente el programa.

Para ello deberemos copiar el listado ensamblador que aparece a continuación, o bien teclear el programa cargador. Aquéllos que utilicen esta útlima opción, deberán ejecutar el programa una vez copiado, y en caso de que no dé ningún mensaje de error, se deberá salvar de la forma siguiente:

SAVE **"RSX"**, B, $A000, 585

Cuando se dese ejecutar, deberemos cargarlo en memoria de la forma indicada al principio de ese artículo, sin olvidarnos de efectuar la llamada a la dirección $A000, con el fin de inicializar los nuevos comandos.

```
10 ;VARIABLES
20        ORG   #A000
30        LD    BC,TABLA
40        LD    HL,ESPACE
50        JP    #BCD1
60 TABLA:  DEFW  NAME
70        JP    VALFA
80        JP    VNUME
90        JP    IMPAN
100 NAME:  DEFM  "VALF"
110       DEFB  "A"+#80
120       DEFM  "VNUM"
130       DEFB  "E"+#80
140       DEFB  "P"+#80
150       DEFB  0
160 ESPACE: DEFS 4
170 IMPAN: LD    A,(PRESOR)
180       CPL
190       LD    (PRESOR),A
200       RET
210 VALFA: AND   A
220       JP    Z,ALFA
230       LD    L,(IX+0)
240       LD    H,(IX+1)
250       LD    A,(HL)
260       INC   HL
270       LD    E,(HL)
280       INC   HL
290       LD    D,(HL)
300       EX    DE,HL
310       LD    DE,DATALF+3
320       LD    C,0
330       LD    B,A
340 MOS:   LD    A,(HL)
350       LD    (DE),A
360       INC   HL
370       INC   DE
380       INC   C
390       DJNZ  MOS
400       LD    A,C
410       LD    (LONCA),A
420       DEC   DE
430       LD    A,(DE)
440       ADD   A,128
450       LD    (DE),A
460       CALL  BUSALF
470       RET
480 VNUME: AND   A
490       JP    Z,NUMER
500       LD    L,(IX+0)
510       LD    H,(IX+1)
520       LD    A,(HL)
530       INC   HL
540       LD    E,(HL)
550       INC   HL
560       LD    D,(HL)
570       EX    DE,HL
580       LD    DE,DATNUM+3
590       LD    C,0
600       LD    B,A
610 MAS:   LD    A,(HL)
620       LD    (DE),A
630       INC   HL
640       INC   DE
650       INC   C
660       DJNZ  MAS
670       LD    A,C
680       LD    (LONCA),A
690       DEC   DE
700       LD    A,(DE)
710       ADD   A,128
720       LD    (DE),A
730       CALL  BUSNUM
740       RET
750 ;
760 BUSALF: LD   HL,DATALF
770       LD    (VARS),HL
780       LD    A,3
790       LD    (VARI),A
800       LD    A,(LONCA)
810       ADD   A,3
820       LD    (LONCV),A
830       CALL  INIC
840       RET
850 BUSNUM: LD   HL,DATNUM
860       LD    (VARS),HL
870       LD    A,2
880       LD    (VARI),A
890       LD    A,(LONCA)
900       ADD   A,3
910       LD    (LONCV),A
920       CALL  INIC
930       RET
940 ALFA:  LD    HL,VARALF
950       LD    (VARS),HL
960       XOR   A
970       LD    (VARI),A
980       LD    A,3
990       LD    (LONCV),A
1000      CALL  INIC
1010      RET
1020 NUMER: LD   HL,VARNUM
1030      LD    (VARS),HL
1040      LD    A,1
1050      LD    (VARI),A
1060      LD    A,3
1070      LD    (LONCV),A
1080      CALL  INIC
1090      RET
1100 INIC:  LD   IX,(VARS)
1110      LD    HL,#170
1120      LD    (DIREC),HL
1130 BUCL:  LD   B,(HL)
1140      INC   HL
1150      LD    C,(HL)
1160      DEC   HL
1170      LD    A,B
1180      OR    C
1190      RET   Z
1200      CALL  BUSC
1210      LD    HL,(DIREC)
1220      LD    DE,(LONLIN)
1230      ADD   HL,DE
1240      LD    (DIREC),HL
1250      JR    BUCL
1260 BUSC:  LD   C,(HL)
1270      INC   HL
1280      LD    B,(HL)
1290      INC   HL
1300      LD    (LONLIN),BC
1310      LD    E,(HL)
1320      INC   HL
1330      LD    D,(HL)
1340      INC   HL
1350      LD    (NUMLIN),DE
1360      XOR   A
1370      LD    (CONCAN),A
1380      LD    D,0
1390 BUC:   LD   A,(LONCV)
1400      CP    D
1410      CALL  Z,FIN
1420      LD    A,(HL)
1430      CP    (IX)
1440      INC   HL
1450      JR    Z,INCCO
1460      LD    D,0
1470      LD    IX,(VARS)
1480      JR    NOINC
1490 INCCO: INC  D
1500      INC   IX
1510 NOINC: DEC  BC
1520      LD    A,B
1530      OR    C
1540      JR    NZ,BUC
1550      RET
1560 FIN:   LD   A,(CONCAN)
1570      AND   A
1580      JR    NZ,FINU
1590      LD    A,13
1600      CALL  IMPRE
1610      LD    A,10
1620      CALL  IMPRE
1630      PUSH  HL
1640      CALL  DECIMA
1650      POP   HL
1660 FINU:  LD   A,(VARI)
1670      CP    0
1680      JR    Z,NOBUS
1690      CP    1
1700      JR    Z,NOBUS
1710      LD    A,(LONCA)
1720      LD    B,A
1730 RESBU: DEC  HL
1740      DJNZ  RESBU
1750 NOBUS: LD   A,(HL)
1760      BIT   7,A
1770      JR    NZ,FINIM
1780      CALL  IMPRE
1790      INC   HL
1800      JR    NOBUS
1810 FINIM: RES  7,A
1820      CALL  IMPRE
1830      LD    A,(VARI)
1840      CP    1
1850      JR    Z,NOALF
1860      CP    2
1870      JR    Z,NOALF
1880      LD    A,"$"
1890      CALL  IMPRE
1900 NOALF: LD   A," "
1910      CALL  IMPRE
1920      LD    IX,(VARS)
1930      LD    D,0
1940      LD    A,(CONCAN)
1950      INC   A
1960      LD    (CONCAN),A
1970      RET
1980 VARALF: DEFB 3,0,0
1990 VARNUM: DEFB 13,0,0
2000 LONCV:  DEFB 3
2010 LONLIN: DEFS 2
2020 NUMLIN: DEFS 2
2030 FINPRO: DEFS 2
2040 DIREC:  DEFS 2
2050 CONCAN: DEFB 0
2060 VARS:   DEFS 2
2070 VARI:   DEFB 0
2080 LONCA:  DEFB 0
2090 DATNUM: DEFB 13,0,0
2100         DEFS 20
2110 DATALF: DEFB 3,0,0
2120         DEFS 20
2130 ;
2140 ;
2150 ;IMPRIME NUMEROS DECIMALES
2160 ;
```

```
2170 ;
2180 DECIMA: SCF
2190         LD   HL,(NUMLIN)
2200         LD   DE,10000
2210         INC  HL
2220         LD   A,47
2230 DMIL:   INC  A
2240         SBC  HL,DE
2250         JR   NC,DMIL
2260         CALL PRINT
2270         LD   DE,1000
2280 MIL:    INC  A
2290         SBC  HL,DE
2300         JR   NC,MIL
2310         CALL PRINT
2320         LD   DE,100
2330 CIEN:   INC  A
2340         SBC  HL,DE
2350         JR   NC,CIEN
2360         CALL PRINT
2370         LD   DE,10
2380 DIEZ:   INC  A
2390         SBC  HL,DE
2400         JR   NC,DIEZ
2410         CALL PRINT
2420         ADD  A,L
2430         CALL PRINT
2440         LD   A," "
2450         CALL IMPRE
2460         RET
2470 PRINT:  CALL IMPRE
2480         LD   A,47
2490         JR   NZ,PAS
2500         INC  HL
2510 PAS:    ADD  HL,DE
2520         INC  HL
2530         RET
2540 IMPRE:  PUSH AF
2550         LD   A,(PRESOR)
2560         AND  A
2570         JR   Z,PANT
2580 WAIT:   CALL #BD2E
2590         JR   C,WAIT
2600         POP  AF
2610         CALL #BD31
2620         RET
2630 PANT:   POP  AF
2640         CALL #BB5A
2650         PUSH HL
2660         PUSH DE
2670         PUSH BC
2680         PUSH IX
2690         PUSH AF
2700         LD   A,66
2710         CALL #BB1E
2720         CALL NZ,PAUSA
2730         POP  AF
2740         POP  IX
2750         POP  BC
2760         POP  DE
2770         POP  HL
2780         RET
2790 PRESOR: DEFB 0
2800 PAUSA:  LD   BC,30000
2810 PAUS:   DEC  BC
2820         LD   A,B
2830         OR   C
2840         JR   NZ,PAUS
2850         CALL #BB03
2860         CALL #BB18
2870         RET
```

```
ALFA    A0AE  BUC     A10C  BUCL    A0DF
BUSALF  A080  BUSC    A0F6  BUSNUM  A097
CIEN    A1E7  CONCAN  A194  DATALF  A1B0
DATNUM  A199  DECIMA  A1C7  DIEZ    A1F2
DIREC   A192  DMIL    A1D1  ESPACE  A020
FIN     A12B  FINIM   A15D  FINPRO  A190
FINU    A140  IMPAN   A024  IMPRE   A20F
INCCO   A122  INIC    A0D5  LONCA   A198
LONCV   A18B  LONLIN  A18C  MAS     A06C
MIL     A1DC  MOS     A042  NAME    A014
NOALF   A172  NOBUS   A152  NOINC   A125
NUMER   A0C1  NUMLIN  A18E  PANT    A220
PAS     A20C  PAUS    A23D  PAUSA   A23A
PRESOR  A239  PRINT   A204  RESBU   A14F
TABLA   A009  VALFA   A02C  VARALF  A185
VARI    A197  VARNUM  A188  VARS    A195
VNUME   A056  WAIT    A216

Table used:    600  from  1000
```

```
10 REM LISTADOR DE VARIABLES
20 REM PROGRAMA CARGADOR
30 FOR N=&A000 TO &A249
40 READ A:SUMA=SUMA+A
50 POKE N,A
60 NEXT
70 IF SUMA<>58826 THEN PRINT "ERROR
 EN DATAS"
80 DATA 1,9,160,33,32,160,195
90 DATA 209,188,20,160,195,44,160
100 DATA 195,86,160,195,36,160,86
110 DATA 65,76,70,193,86,78,85
120 DATA 77,197,208,0,0,0,0
130 DATA 0,58,57,162,47,50,57
140 DATA 162,201,167,202,174,160,22
1
150 DATA 110,0,221,102,1,126,35
160 DATA 94,35,86,235,17,179,161
170 DATA 14,0,71,126,18,35,19
180 DATA 12,16,249,121,50,152,161
190 DATA 27,26,198,128,18,205,128
200 DATA 160,201,167,202,193,160,22
1
210 DATA 110,0,221,102,1,126,35
220 DATA 94,35,86,235,17,156,161
230 DATA 14,0,71,126,18,35,19
240 DATA 12,16,249,121,50,152,161
250 DATA 27,26,198,128,18,205,151
260 DATA 160,201,33,176,161,34,149
270 DATA 161,62,3,50,151,161,58
280 DATA 152,161,198,3,50,139,161
290 DATA 205,213,160,201,33,153,161
300 DATA 34,149,161,62,2,50,151
310 DATA 161,58,152,161,198,3,50
320 DATA 139,161,205,213,160,201,33
330 DATA 133,161,34,149,161,175,50
340 DATA 151,161,62,3,50,139,161
350 DATA 205,213,160,201,33,136,161
360 DATA 34,149,161,62,1,50,151
370 DATA 161,62,3,50,139,161,205
380 DATA 213,160,201,221,42,149,161
390 DATA 33,112,1,34,146,161,70
400 DATA 35,78,43,120,177,200,205
410 DATA 246,160,42,146,161,237,91
420 DATA 140,161,25,34,146,161,24
430 DATA 233,78,35,70,35,237,67
440 DATA 140,161,94,35,86,35,237
450 DATA 83,142,161,175,50,148,161
460 DATA 22,0,58,139,161,186,204
470 DATA 43,161,126,221,190,0,35
480 DATA 40,8,22,0,221,42,149
490 DATA 161,24,3,20,221,35,11
500 DATA 120,177,32,226,201,58,148
510 DATA 161,167,32,15,62,13,205
520 DATA 15,162,62,10,205,15,162
530 DATA 229,205,199,161,225,58,151
540 DATA 161,254,0,40,11,254,1
550 DATA 40,7,58,152,161,71,43
560 DATA 16,253,126,203,127,32,6
570 DATA 205,15,162,35,24,245,203
580 DATA 191,205,15,162,58,151,161
590 DATA 254,1,40,9,254,2,40
600 DATA 5,62,36,205,15,162,62
610 DATA 32,205,15,162,221,42,149
620 DATA 161,22,0,58,148,161,60
630 DATA 50,148,161,201,3,0,0
640 DATA 13,0,0,3,0,0,0
650 DATA 0,0,0,0,0,0,0
660 DATA 0,0,0,13,0,0,0
670 DATA 0,0,0,0,0,0,0
680 DATA 0,0,0,0,0,0,0
690 DATA 0,0,0,0,0,3,0
700 DATA 0,0,0,0,0,0,0
710 DATA 0,0,0,0,0,0,0
720 DATA 0,0,0,0,0,0,0
730 DATA 55,42,142,161,17,16,39
740 DATA 35,62,47,60,237,82,48
750 DATA 251,205,4,162,17,232,3
760 DATA 60,237,82,48,251,205,4
770 DATA 162,17,100,0,60,237,82
780 DATA 48,251,205,4,162,17,10
790 DATA 0,60,237,82,48,251,205
800 DATA 4,162,133,205,4,162,62
810 DATA 32,205,15,162,201,205,15
820 DATA 162,62,47,32,1,35,25
830 DATA 35,201,245,58,57,162,167
840 DATA 40,10,205,46,189,56,251
850 DATA 241,205,49,189,201,241,205
860 DATA 90,187,229,213,197,221,229
870 DATA 245,62,66,205,30,187,196
880 DATA 58,162,241,221,225,193,209
890 DATA 225,201,0,1,48,117,11
900 DATA 120,177,32,251,205,3,187
910 DATA 205,24,187,201,0,0,0
```

Para que tus dedos no realicen el trabajo duro, M.H. AMSTRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicítanoslo.

AMSTRAD CPC - 464 AMSTRAD



ORDENADORI

SERIE CPC

• **TECLADO** • Teclado profesional con 74 teclas en 3 bloques - Hasta 32 teclas programables - Teclado redefinible
• **PANTALLA** • Monitor RGB verde (12") o color (14")

	Normal	Alta Res.	Multicolor
Col × líneas	40 × 25	80 × 25	20 × 25
Colores	4 de 27	2 de 27	16 de 27
Puntos	320 × 200	640 × 200	160 × 200

– Se pueden definir hasta 8 ventanas de texto y 1 de gráficos • **SONIDO**
• 3 canales de 8 octavas moduladas independientemente - Altavoz interno regulable - Salida estéreo • **BASIC**
• Locomotive BASIC ampliado en ROM - Incluye los comandos AFTER y EVERY para control de interrupciones

AMSTRAD CPC 464

UNIDAD CENTRAL. MEMORIAS
• Microprocesador Z80A - 64K RAM ampliables - 32K ROM ampliables
CASSETTE • Cassette incorporada con velocidad de grabación (1 o 2 Kbaudios) controlada desde Basic • **CONECTORES**
• Bus PCB multiuso, Unidad de Disco exterior, paralelo Centronics, salida estéreo, joystick, lápiz óptico, etc.
• **SUMINISTRO** • Ordenador con monitor verde o color - 8 cassettes con programas - Libro "Guía de Referencia BASIC para el programador" - Manual en castellano - Garantía Oficial AMSTRAD ESPAÑA.

TODO POR	**59.900 Pts.** (monitor verde)
	90.900 Pts. (monitor color)

AMSTRAD CPC 6128

UNIDAD CENTRAL. MEMORIAS
• Microprocesador Z80A - 128 K RAM ampliables - 48 K ROM ampliables
UNIDAD DE DISCO • Unidad incorporada para disco de 3" con 180K por cara • **SISTEMAS OPERATIVOS**
• AMSDOS, CP/M 2.2, CP/M Plus (3.0)
• **CONECTORES** • Bus PCB multiuso, paralelo Centronics, cassette exterior, 2.ª Unidad de Disco, salida estéreo, joysticks, lápiz óptico, etc.
• **SUMINISTRO** • Ordenador con monitor verde o color - Disco con CP/M 2.2 y lenguaje DR. LOGO - Disco con CP/M Plus y utilidades - Disco con 6 programas de obsequio - Manual en castellano - Garantía Oficial AMSTRAD ESPAÑA.

TODO POR	**84.900 Pts.** (monitor verde)
	119.900 Pts. (monitor color)

PCW - 8256 AMSTRAD CPC - 6128

ES AMSTRAD ¡¡Increíble!!

AMSTRAD PCW 8256

UNIDAD CENTRAL. MEMORIAS
• Microprocesador Z80A - 256K RAM de las que 112K se utilizan como disco RAM **• TECLADO •** Teclado profesional en castellano (ñ, acento...) de 82 teclas **• PANTALLA •** Monitor verde de alta resolución - 90 columnas × 32 líneas de texto **• UNIDAD DE DISCO •** Disco de 3" y 173K por cara - Opcionalmente, 2.ª Unidad de Disco de 1 Mbyte integrable **• SISTEMA OPERATIVO •** CP/M Plus de Digital Research **• IMPRESORA •** Alta calidad (NLQ) a 20 c.p.s. - Calidad estándar a 90 c.p.s. - Papel continuo u hojas sueltas - Alineación automática del papel - Caracteres normales, comprimidos, expandidos, control del paso de letra (normal, cursiva, negrita, subíndices, superíndices, subrayado, etc). **• OPCIONES •** Kit de Ampliación a 512K RAM y 2.ª Unidad de Disco - Interface Serie RS 232C y paralelo Centronics **• SUMINISTRO •** Ordenador completo con teclado, pantalla, Unidad de Disco e Impresora - Discos con el procesador de Texto LocoScript, CP/M Plus, Mallard BASIC, DR. LOGO y diversas utilidades - Manuales en castellano - Garantía Oficial AMSTRAD ESPAÑA.

TODO POR 129.900 Pts.



Los más prestigiosos paquetes de **Software Profesional,** en formato AMSTRAD... a **"precios AMSTRAD"**

Existe también la versión **PCW 8512** con **512K RAM** y la 2.ª Unidad de Disco de 1 Mbyte incorporada. **PVP. 149.900 Pts.**
* El **PCW 8256** puede utilizarse como terminal y en comunicaciones.

El I.V.A. no está incluido en los precios.

NOTA: Es muy importante verificar la garantía del aparato ya que sólo **AMSTRAD ESPAÑA** puede garantizarle la ordenada reparación y sobre todo materiales de repuesto oficiales (Monitor, ordenador, cassette o unidades de discos).

AMSTRAD ESPAÑA

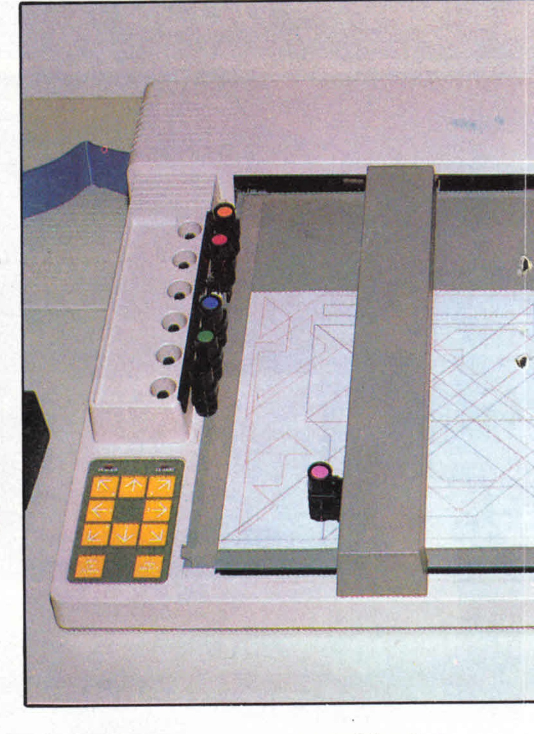
C/. Aravaca, 22. Tel. 459 30 01. Télex 47660 INSC E. Fax 459 22 92. 28040 Madrid.

Delegación en Cataluña: C/. Tarragona, 110. Tel. 325 10 58. 08015 Barcelona.

EL MUNDO DE LOS «PLOTTERS»

Francisco G.R.

Cuando realizamos unos gráficos y deseamos observarlos desde una perspectiva no tan horizontal, la realizada por el usuario ante el monitor, nos planteamos la necesidad de volcarlo en papel. Esto se puede realizar en una impresora mediante pequeñas rutinas en máquina que nos realizan esta labor.



Al principio esta solución en válida para un tanto por ciento muy importantes de usuarios. Pero, poco a poco, vamos comprobando que no es una reproducción muy perfecta la que obtenemos en la impresora. Hay varios puntos muy claros en este aspecto: entre ellos podemos citar las circunferencias, que no son tales sino más bien elipses, y luego tenemos las intersecciones o las líneas rectas que enlazan un par de puntos. Los enlaces con los puntos son perfectos, pero la trayectoria de la recta deja mucho que desear con la realidad.

Esto y la necesidad de adaptación a necesidades concretas nos obligan a buscar un periférico de impresión más específico: el PLOTTER. Este periférico está diseñado para trasladar al papel el desarrollo de un plano (en construcción), del diseño de piezas especializadas, de diseño cartográfico, etc.

Salidas por el canal de impresión

En un ordenador hay que tener en cuenta las entradas y salidas de datos por los diferentes canales de la máquina. Como ejemplo, podemos citar que, cuando estamos introduciendo datos por el teclado, éstos se imprimen en la pantalla por el canal de vídeo. Lo mismo sucede cuando mandamos información hacia la impresora o el Plotter: la llevamos por su canal. Este es el encargado de transmitir la información contenida en la pantalla, o en un fichero, hacia el periférico que tengamos conectado.

Comenzamos tratando las rutas que comunican el procesador y los módulos de entrada y salida de un microordenador. Para ello, empezamos con una introducción sobre las diferencias de comunicación Serie y Paralelo.

Comunicación en Serie

Especialmente cuando se trata de transmitir datos a larga distancia, la comunicación en Paralelo resultaría inviable por el número de hilos necesario para la misma. Por ello, recurrimos a la transmisión en Serie, en la que los bit se suceden ordenadamente en el tiempo; sólo son estrictamente necesarios dos hilos activos de interconexión, uno para salida y otro para entrada de datos.

La diferencia fundamental entre los interfaces para la transmisión en Serie y en Paralelo es que las series deben realizar la conversión del dato de Paralelo a Serie para la emisión, y viceversa para la recepción.

Un ejemplo de un interface Serie es el RS-232C, cuyas siglas significan Recomended Standard. En este interface, además de llevar los dos hilos de interconexión, se dispone de una serie de señales que permite controlar la transmisión.

Comunicación en paralelo

Al contrario de lo que sucede con los interfaces Serie, los paralelos tienen la ventaja de poder realizar transferencias de datos a muy alta velocidad, ya que se transmiten los caracteres en una sola emisión. La desventaja fundamental es el mayor número de hilos de interconexión y la corta distancia que pueden cubrir, no sólo en razón de su costo y complejidad, sino porque no son aptas para las comunicaciones por vía telefónica.

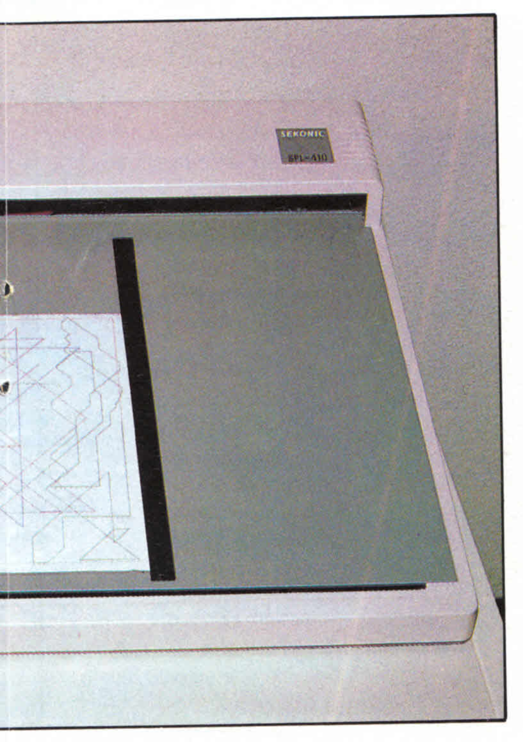
Los interfaces paralelos, por no tener la necesidad de adaptarse a un medio común (en el caso de los interfaces en Serie las líneas telefónicas eran este medio común), no se'han sujetado por lo general a ningún standard, y esto provoca que frecuentemente existan problemas de incompatibilidades. Mediante una norma que se elaboró, llamada IEEE 488 y que incorporan la mayoría de los equipos actualmente, aunque aún no todos, se intentó solucionar el problema de las incompatibilidades.

Programas con opción de Plotter en el mercado

Cuando decidimos comprar un Plotter, miramos las posibilidades de resolución que tiene éste, y el tamaño del folio que acepta. También procuramos que disponga del mayor número de plumas de colores posibles, para darle una mayor calidad a los trabajos que realicemos.

El primer paso a seguir será la adaptación de éste a nuestra máquina. Aunque dependiendo de la calidad del Plotter, se puede o no controlar manualmente, tenemos la necesidad de gobernarlo a través de nuestro equipo. Si comenzamos por estudiar los puertos de entrada y salida y cómo mandar la información, probablemente no utilizaríamos el Plotter desde el ordenador.

Para ello nos vamos a programas ya comercializados y que nos den de alguna forma este problema resuelto. Si buscamos y miramos

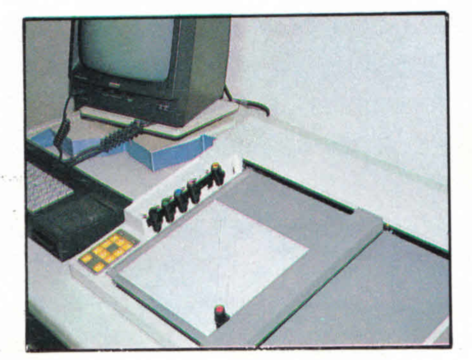


bien los programas que existen, comprobaremos que hay dos programas en el que se da la opción de salida por Plotter. Estos programas son el D.R. Draw y D.R. Grahp, los cuales en las opciones de imprimir tienen la salida de Plotter.

Cuando realizamos un dibujo con el D.R. Draw, una vez retocado, lo podemos mandar por la salida de Plotter, pero podemos encontrarnos con la respuesta de «programa no preparado para salidas de ese periférico», o bien no recibir ninguna señal el Plotter. El mismo caso ocurre con el D.R. Graph: tenemos que preparar la salida del Plotter antes de su utilización.

La inicialización del programa para el ordenador de que dispongamos viene explicada en el manual, pero ya no es tan clara para instalar el Plotter. Lo primero que debemos tener en cuenta es el interface que lleva incorporado nuestro Plotter. Dependiendo de ésta tenemos que obrar de una forma o de otra.

Si tuviera un interface Serie, lo primero que necesitaríamos, tanto para la gama de los CPC o de los PCW, sería la conexión de un interface Serie para el equipo. Este interface lo conectaremos a la salida que llevan nuestros CPC (hay que hacer constar que en el 6128, se conecta en el Bus de salida impresora, mientras que en el 464 y los PCW 8256 y 8512, se conectan al Bus trasero de expansión).



En caso contrario al anterior (interface Paralelo), la cosa se reduce ha adquirir un cable Centronics y conectarlo directamente al equipo. Esto se puede hacer con la gama de los CPC. Desgraciadamente para los usuarios de los últimos modelos lanzados de **Amstrad,** hay que decir que obligatoriamente necesitan la compra del interface Paralelo, por carecer este equipo de una salida Centronics.

Cuando hemos solucionado todos los problemas de tipo Hardware, de conexión entre ambos, tenemos que instalar el Software. Con la compra de nuestro ordenador nos dan unos discos de Sistemas Operativos, en los cuales también existen unas utilidades para la máquina. Efectivamente, uno de esos programas que hemos sacado por el directorio tantas veces y no sabemos para qué nos sirve y del cual no disponemos de ninguna información, es ahora mismo el programa más importante para la instalación.

Cuando instalemos el programa para nuestro equipo, debemos incorporar la salida pertinente. Lo realizaremos colocando en el disco los DEVICE de OUTPUT de SID (cuando tenemos conectados un interface en Serie), o CEN (caso contrario al anterior, el interface es Paralelo o Centronics). Cuando se trata de una salida en Paralelo, en los modelos **Amstrad** CPC no hace falta preparar la salida, pues están tomados estos valores por defecto. En el caso de Serie sí son necesarios, debido a tener que incorporarles al equipo un periférico externo.

Una vez preparados los DEVICE CEN o DEVICE SID, el ordenador tiene conectados los canales de impresión para su uso. Seguidamente debemos introducir en el disco el programa con el cual el ordenador mande señales concretas y admitidas por el Plotter. Este programa nos viene con el equipo y se denomina:

A > DDHP7470.PRL

De él existen dos versiones dependiendo del Sistema Operativo y del equipo. Existe una en el disco número dos del CP/M Plus del 6128, y la otra se encuentra en la cara número tres de las utilidades del CP/M Plus del 8256/8512. Este programa debe colocarse inmediatamente después de los DEVICE, para su perfecto funcionamiento.

Comandos de manejo de Plotter

Al igual que en una impresora, un Plotter tiene sus comandos de uso interno. En una impresora tenemos comandos de salto de página, retroceso de carro y salto de línea, entre otros.

En un Plotter estos comandos no existen, pero sí tenemos otros, como pueden ser el cambio de tinta, posicionamiento en un punto del papel, radio de una circunferencia, etc.

Como punto de interés por el usuario, vamos a definir la mayoría de los comandos del Plotter para los intrépidos que deseen hacer uso de ellos.

Comando: AA — Arc Absolute (Arco Absoluto)
— Formato: AA x,y,a(,c)
x = Punto de eje x. a = Grados
y = Punto eje y. c = Grados de retroceso

Comando: AR — Arc Relative (Arco Relativo)
— Formato: AR ix,iy,a(,c)
ix = Incremento de x. a = Grados
iy = Incremento de y. c = Grados de retroceso

Comando: CI —CIRCLE (Circunferencia)
— Formato: CIr (,c)
c = Grados

Comando: LB — Label (Etiqueta)
— Formato: LB c1,c2,... cN[t]
cN = Punto de comienzo, alto, ancho, etc.
t = Espacio entre etiqueta

Comando: PU — Pen Up (Pluma Arriba)
— Formato: PU;

Comando: LB — Label (Etiqueta)
— Formato: LB c1,c2,... cN[t]

cN = Punto de comienzo, alto, ancho, etc.
t = Espacio entre etiqueta

Comando: PU — Pen Up (Pluma Arriba)
— Formato: PU;

Comando: PD — Pen Down (Pluma Abajo)
— Formato: PD;

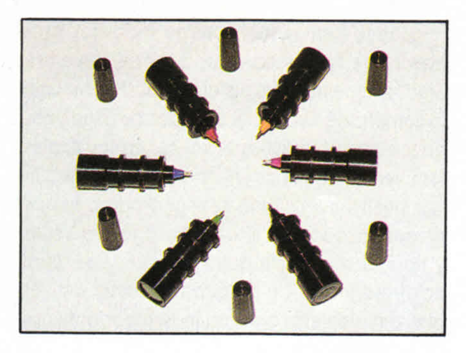
Comando: PA — Plot Absolute (Plot Absoluto)
— Formato: PA;

Comando: PR — Plot Relative (Plot Relativo)
— Formato: PR;

Comando: FT — Fill Type (Relleno de Figura)
— Formato: FT;

Comando: RA — Rectangle Absolute (Rectángulo Absoluto)
— Formato: RA x,y
x = Punto de eje x. y Punto eje y.

Comando: CS — Designate Standard Character Set (Definir Carác.)
— Formato: CSn
n = Número de Set de caracteres



Comando: CA — Designate Alternate Character Set (Definir Carác.)
— Formato: CSn;
n = Número de Set creado por usuario

Comando: SS — Select Standard Set (Seleccionar un Standard)
— Formato: SS,

Comando: SA — Select Alternate Set (Seleccionar uno propio)
— Formato: SA;

Comando: DT — Define Terminator (Definir el término)
— Formato: DT c;
c = valor del punto

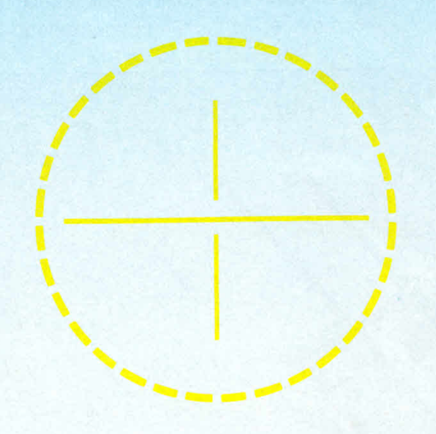
Comando: DI — Absolute Direction (Dirección Absoluta)
— Formato: DI ix,iy ;
ix = Incremento en la dirección X
iy = Incremento en la dirección Y

Comando: DR — Relative Direction (Dirección Relativa)
— Formato: DR ix, iy ;
ix = Incremento en la dirección X
iy = Incremento en la dirección Y

Comando: SL — Character Slant (Elongación de carácter)
— Formato: SL c ;
c = Grados de elongación

SIMULADORES DE COMBATE

De todas las aventuras que podemos vivir sobre el teclado de un ordenador, tal vez una de las más reales y emocionantes sea la de pilotar un avión de caza en busca de aparatos enemigos.



Uno de los clásicos en los programas de juego para ordenador, son los simuladores de vuelo. Su nacimiento data de hace tiempo, cuando los ordenadores personales sólo se utilizaban para negocios.

Estos programas estaban destinados a llenar los ratos libres de ejecutivos. Mientras se tomaba el café después de una reunión, surgía el momento idóneo para lucir ante los demás nuestras habilidades como piloto.

En un momento la pantalla del ordenador se convertía en el centro de atención, y las palabras de ánimo de los camaradas llenaban la sala.

Con la llegada de los ordenadoles domésticos, y la aparición de un elevado número de clientes potenciales de los juegos, ha llegado también la producción masiva de software de entretenimiento, y cómo no, los simuladores de vuelo han tenido un lugar destacado en esta invasión de productos.

Desde las primeras piezas de museo, en las que sólo salía en pantalla la línea del horizonte, y unos cuantos indicadores digitales de altitud, velocidad, etc., hasta los actuales, en los que impresionantes efectos tridimensionales hacen aparecer aviones enemigos que nos lanzan misiles, la cosa ha evolucionado.



Mucho más que tirar del joystick

Dentro del mundo de los juegos, los simuladores de combate ocupan un sitio especial. Nos encontramos ante un tipo de juego que precisamente no es de matar marcianitos, estos programas requieren mucho más que tirar del joystick para su utilización. En ellos nos encontramos a los mandos de un caza, y hemos de interceptar al enemigo, que se encuentra en una posición detectada por el radar en tierra.

Una vez conocida la posición de nuestro enemigo, hemos de reproducir todo el proceso que realizaría un piloto de verdad.

En primer lugar hemos de despegar de la pista, para lo cual encenderemos el motor, accionaremos los flaps, aceleraremos y nos elevaremos.

Cuando hemos cogido altura, es la hora de ocultar el tren de aterrizaje y consultar el mapa para localizar la posición del enemigo.

Si el modelo de simulador es avanzado, el ordenador de abordo nos llevará automáticamente al encuentro con él, si no, tendremos que navegar en su búsqueda, valiéndonos de todos los instrumentos de navegación.

Tras varios minutos de singladura, nos encontramos en la posición del avión a cazar: el mapa nos indica su proximidad y nuestros ojos escudriñan el aire a nuestro alrededor en su búsqueda.

Al fin vemos un punto que se aproxima hacia nosotros; es el momento de armar el sistema de misiles y prepararnos para abordarle.

Con el dedo sobre el botón de disparo, intentamos ponernos a la cola del avión enemigo aumentando la velocidad; cuando nos encontramos tras su estela, una ligera pulsación del disparador y presenciamos la trayectoria del misil tras el reactor del aparato. Unos segundos después la explosión del aparato, nos permite añadir una muesca más a nuestro récord de derribos.

Pero mientras observamos con orgullo cómo caen los restos del caza enemigo, nuestros momentos de falta de concentración, han sido aprovechados por otro caza para colocarse a nuestra cola.

Sólo nos queda intentar una maniobra de defensa, o seremos derribados inmediatamente. Forzando al máximo las posibilidades de nuestro caza, realizamos una tijera para intentar colocarnos a su cola.

Hemos fallado en nuestro intento de evasión, y la situación se complica, intentamos un giro en el aire, pero ya es demasiado tarde, un misil ha sido lanzado y su impacto llega certero con una terrible explosión que inunda de fuego nuestra cabina.

En el espacio de tiempo transcurrido desde nuestro despegue, hasta el contacto con el avión enemigo, hemos tenido que accionar un elevado número de controles, al igual que vi-

gilar el calentamiento de motores, gasto de combustible, controlar el rumbo, etc.

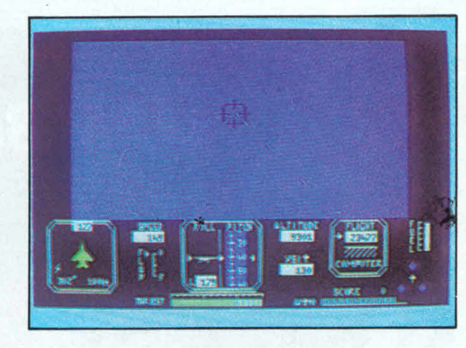
En el combate hemos de situar la mira, aumentar velocidad, colocarnos a la cola del enemigo, armar misiles, tirar de la ametralladora y derribar al otro caza.

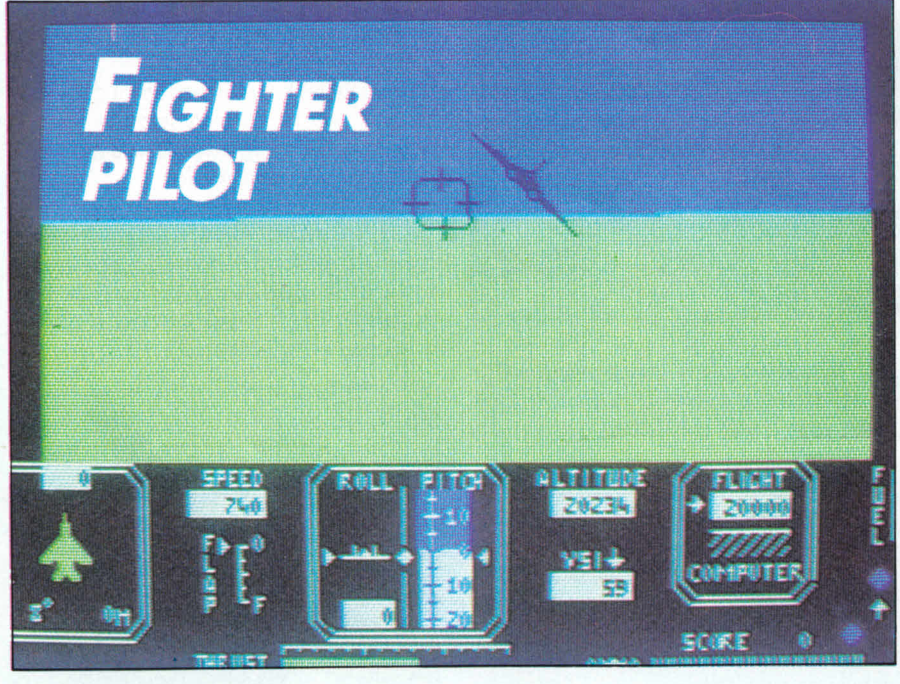
Estos son los factores que hacen de un simulador de combate, un programa mucho más complicado e interesante que una simple matanza de marcianitos.

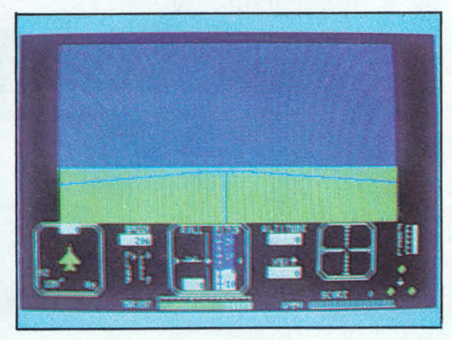
Computadora de vuelo, se utiliza indistintamente para servir de guia en el aterrizaje, así como para el combate aéreo, donde se demuestra su eficacia.

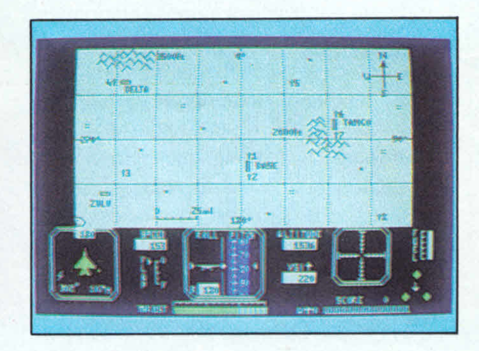
Indicador de aterrizaje, con él podemos comprobar que nuestra maniobra es correcta, midiendo el ángulo de inclinación que llevamos.

Combustible, indicador del tren de aterrizaje y mapa.









El camino de los simuladores de combate en el **Amstrad**, se inicia con el Fihgter Pilot, programa de la casa Digital Integration.

En este caso estamos a los mandos de un F-15, y nos disponemos a interceptar al enemigo.

La cabina de mandos aparece ante nuestros ojos, cuajada de instrumentos e indicadores. Para pilotar nuestro avión disponemos de una amplia gama de controles:

Horizonte artificial, con un avión indicador de la posición y los valores numéricos de los ángulos.

Altímetro e indicador de velocidad.

Indicador de velocidad vertical, con el cual podemos determinar la velocidad de ascenso o descenso de nuestro caza.

Empuje, con indicador gráfico de porcentaje de fuerza de propulsión y calentamiento de motores.

Radar y brújula, con los cuales dirigimos nuestro caza, apoyándonos en las distintas bases de tierra, o siguiendo en el mapa la posición de nuestro enemigo.

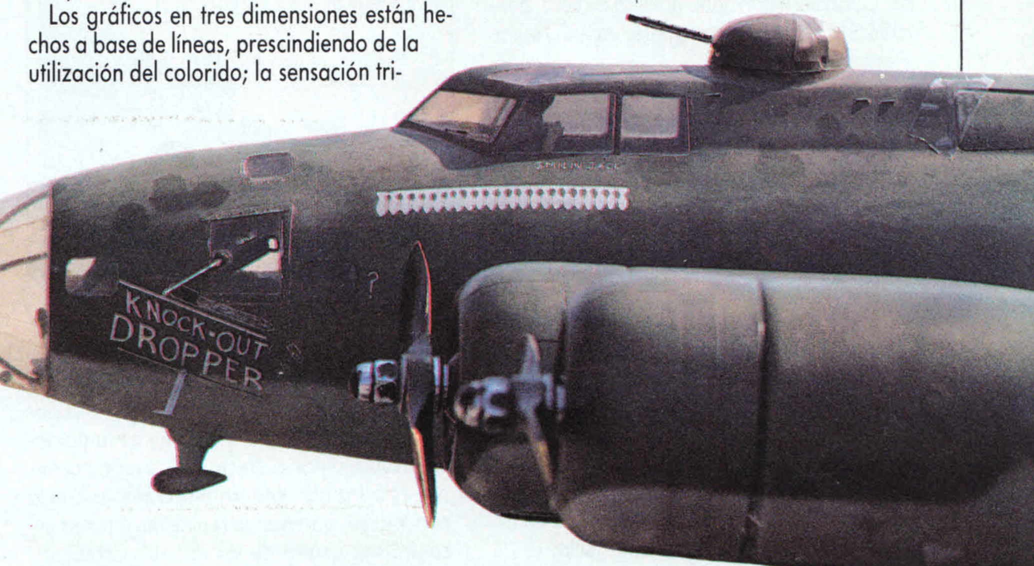
Modo de combate.

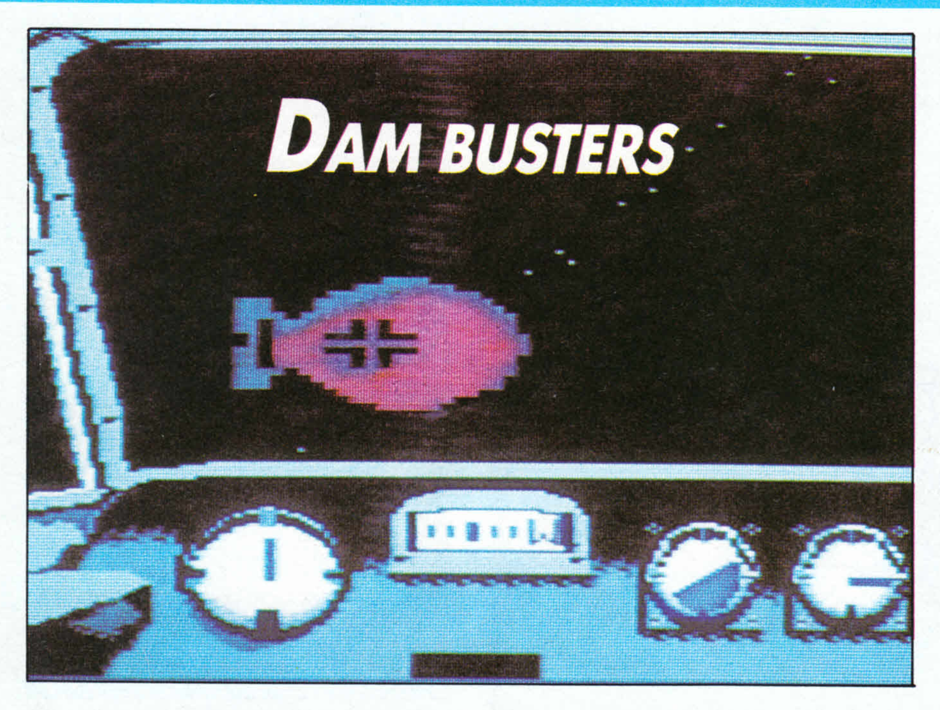
Con él iniciamos la maniobra de aproximación a nuestro blanco. Al ser activado se ponen en funcionamiento el radar y la computadora de vuelo, dándonos los datos exactos de su posición y altitud.

Indudablemente una amplia gama de instrumentos e indicadores, que gobiernan el caza. Para pilotar el F-15 tenemos que manejar junto con el joystick 16 teclas, lo que da idea de la complejidad de la operación de pilotar este aparato. Nos encontramos ante un programa en el que el manejo del avión, debido al elevado número de controles, resulta un poco costoso; en todo momento ante nuestros ojos tenemos presentes los tableros de mandos y a la vista a través de la cabina del piloto.

Los gráficos en tres dimensiones están hechos a base de líneas, prescindiendo de la utilización del colorido; la sensación tridimensional es aceptable, pero poco espectacular.

Un programa con excesivo número de teclas a utilizar, en el cual la navegación se hace lenta y la localización del enemigo bastante laboriosa, recomendado especialmente para los meticulosos de las técnicas de vuelo, que se tendrán que armar de paciencia hasta ponerse a la cola del enemigo.





Todos los programas tratados anteriormente, tenían una característica en común; aunque se tratara de modelos más antiguos o modernos, siempre pilotábamos un caza, el Harrier, el Spitfire, el F-15, o el futurista Skyfox.

Pero como la calenturienta imaginación de los creadores de software no puede estancarse en la vulgaridad, se les ocurre romper con todos los moldes, reproduciendo una acción de guerra real de la Segunda Guerra Mundial.

En primer lugar, ya no pilotamos un caza, sino que estamos a los mandos de un bombardero pesado Lancaster, y como toque distintivo, nuestro objetivo son las presas de la zona industrial del valle del Ruhr, en pleno corazón de la Alemania nazi.

Como se trata de manejar un bombardero pesado, ya no basta con la visión a través de la cabina del piloto, como en los cazas, aquí tenemos una completa dotación de personal con distintos puestos asignados, a los que nosotros debemos sustituir.

Los puestos que hemos de ocupar son los de piloto, artillero principal, artillero de cola, operador de bombas, navegante, ingeniero jefe e ingeniero segundo.

El programa está concebido de forma que el cambio de puesto se realize de forma fácil, y permita el desarrollo paralelo de la misión sin interrupciones.

Cada elemento de la tripulación, tiene asignada una pantalla distinta, de forma que con un simple toque de tecla cambiamos al puesto deseado.

Con este método conseguimos dominar el aparato, solamente con el joystick y el uso de siete teclas, facilitando la complicada tarea que representa hacer las veces de la dotación completa de un Lancaster.

Para romper más aún con la línea clásica de los simuladores de vuelo, Dam Busters introduce un nuevo sistema de pilotaje, que no requiere la pulsación ni de media tecla.

La sala de máquinas, donde se encuentran los mandos que gobiernan los motores del avión, asombrosamente está toda gobernada por el joystick.

Una vez que nos encontramos en la pantalla del ingeniero jefe, aparecen los ocho indicadores de revoluciones y de inyección, y las ocho palancas que los accionan, más las cuatro que sirven para extinguir el fuego en cada uno de los motores.

No hemos de olvidar que el Lancaster está dotado de cuatro motores, cada uno de los cuales se puede manejar por separado.

Para accionar toda esta serie de mandos sólo hemos de hacer uso exclusivo del joystick. Un punto negro aparece debajo del grupo de palancas y para manejar la deseada sólo hemos de mover el punto hasta ella, y una vez situada debajo tirando del joystick hacia arriba o hacia abajo, la palanca se mueve en la misma dirección.

Con este método gráfico de accionamiento de controles, no hay problemas de teclas ni de vigiliar números.

Todo el programa está tratado con unos gráficos excelentes, el dibujo de cada pantalla, ayudado por el efecto de la visión nocturna, está tratado con un hiperrealista efecto perspectivo, pareciendo realmente que nos encontramos dentro de las distintas cabinas.

Un programa con un concepto totalmente nuevo, realizado con excelentes gráficos, y en el que los distintos puestos que hemos de ocupar, y la minuciosidad con que hemos de estudiar el lanzamiento de las bombas rompedoras, le dotan de un interés y emoción insuperables. Su creadora la casa U.S. Gold.

Como podemos ver los usuarios de Amstrad interesados por el mundo de los simuladores de vuelo, tienen una amplia gama de posibilidades para elegir.

Sea cual sea el tipo de programa que más se ajuste a nuestras preferencias, encontraremos uno a nuestra medida.

Si deseamos un simulador de los de la vieja escuela, con largas maniobras de vuelo y aproximación, manejando multitud de controles, tenemos el Fighter Pilot.

Si por el contrario nuestro gusto se dirige hacia maniobras más rápidas, utilizando un número más reducido de controles; aderezadas con unas pantallas de cabina y panel de mandos, de un sobervio realismo gráfico, nuestro programa es el Spitfire 40.

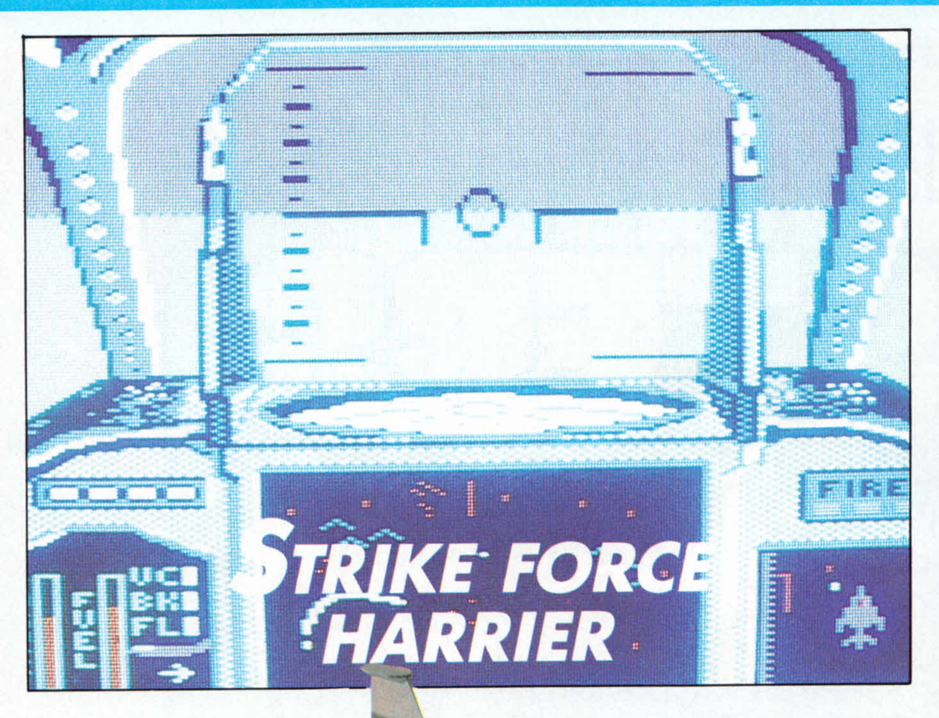


Para los amantes de aviones más modernos y sotisficados, en los que hemos de manejar una gran cantidad de controles, y en los que la misión pasa a ser estratégica, incluyendo combates tierra aire y aire aire, la solución idónea es el Strike Force Harrier.

Los que no quieran complicarse la vida con multitud de teclas y largas aproximaciones de vuelo, lo tienen muy fácil: el Skyfox ofrece la posibilidad de aniquilar a base de misiles o ametralladora; tanques y escuadrillas enemigas, en un programa de acción total y los mejores efectos tridimensionales.

Los que en cambio quieran reproducir el curso de la historia, se econtrarán de lleno en la misión más audez de la Segunda Guerra Mundial, a los mandos de un bombardero Lancaster, en un programa en que todo ha sido hecho para manejarse sin esfuerzo, con excelentes gráficos y en el que las emociones de una incursión nocturna se saborean hasta el final.

Después de ver todo esto, ¿quién no se pregunta cómo serán los nuevos simuladores de vuelo?



STRIKE FORCE HARRIER

El protagonista de esta nueva pieza de software, es el famoso Harrier, un avión mucho más próximo a nosotros que el anterior, y con la especial característica de que sus motores orientables, le permiten despegar y aterrizar en vertical.

El programa es de la misma casa que el anterior, por lo cual a primera vista ya sabemos que encontraremos un producto de una calidad inmejorable.

Si hay algo que impresiona al abrir la carpeta de Harrier, es el librito de instrucciones, que contiene 26 páginas, las cuales están cuajadas de explicaciones de los instrumentos que manejamos, técnicas de vuelo, controles, descripción de la misión, y una completa guía ilustrada de las distintas técnicas de defensa y ataque en vuelo que permite nuestro Harrier.



En la cabina de mandos tenemos los siguientes controles: VAV dispositivo de información de vuelo. Con él podemos obtener los datos de velocidad vertical, velocidad relativa, girocompás, altura sobre el nivel del suelo y cabeceo del avión respecto a la horizontal.

Mira de bombas, con el punto de posible impacto si se suelta una bomba.

Localización de puntos de aterrizaje, donde el personal de tierra puede abastecernos.

Mira de misil.

Visualización multifuncional; con información del vuelo y estado del armamento, potencia de empuje, vector de empuje, alimentación de combustible, posición de flaps, tren de aterrizaje y frenos.

Radar de ataque aéreo; que detecta en un radio de cinco millas la posición tanto de aviones enemigos, como de misiles SAM y radar de identificación y seguimiento; con él podemos reconocer el terreno que se encuentra dentro del área de operaciones, en él se identifican montañas, bases de tierra, pistas de aterrizaje, bases de misiles SAM, tanques, aviones y misiles enemigos.

Por último temenos la pantalla de mensajes, que es la terminal del ordenador de vuelo, en la que se nos da información de inestimable valor.

Sin ninguna duda estamos ante el más completo y sofisticado de los simuladores tratados hasta ahora. Las características del Harrier con sus motores orientables y el concepto totalmente nuevo de este avión de combate, hacen que su conducción requiera una técnica distinta a la de los aviones convencionales.

El número de teclas que accionan los distintos controles y dispositivos es de 20 más el joystick; un completo panel de mandos.

Este programa, además de las características de simulación de vuelo, introduce una nueva variante en los simuladores y nuestra misión no sólo se limita a derribar aparatos enemigos, sino que se trata de una completa incursión en su territorio.

Nuestro objetivo es su Cuartel General, que se encuentra a 500 millas de nuestra posición. Para llegar hasta él, hemos de combatir a las fuerzas acorazadas enemigas que hostigan nuestros puntos de apoyo en tierra, las cuales nos servirán para reabastecer las deficiencias de armamento y combustible.

Aparte de los combates aire tierra, también tenemos que eliminar a los cazas MIG-26, que patrullan los aires en nuestra búsqueda.

Harrier es un completo simulador de vuelo, dotado de buenos gráficos tridimensionales realizados a todo color, y en el que se introducen conceptos de misión completa, en la que la estrategia y el aprovechamiento de las excepcionales características de nuestro caza marcan la diferencia.

Otro gran programa de Mirrorosoft.

JUEGA EL JUEGO DEL QUE TODOS HABLAN !

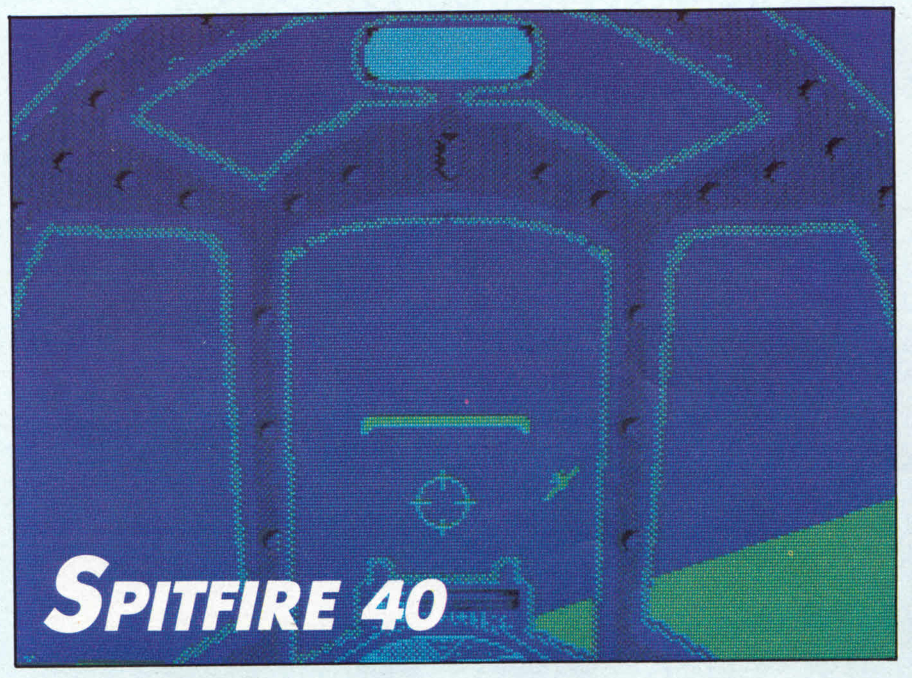


¡Apasiónante!

Son las 21'15 horas del 16 de Mayo. Un bombardero Lancaster en vuelo especial, despega de Inglaterra hacia Alemania. Después de meses de preparación, el escuadrón 617 vuela en una operación destinada a cambiar el curso de la II Guerra Mundial. Su objetivo es destruir las más importantes presas alemanas para paralizar los puntos vitales de sus fábricas de armamento.

Este detallado y auténtico simulador te permite ocupar los puestos de: **Piloto, Ingeniero de vuelo, Artillero delantero y trasero, Bombardero** y **Navegante.** Volarás a través del Canal de la Mancha y Europa intentando evitar a los temibles ME-110 alemanes, zeppelines, focos antiaéreos y todos los demás peligros a los que se enfrentó el comando Inglés.

PIDE ESTOS PROGRAMAS A ERBE, SANTA ENGRACIA, 17, 28010 MADRID. TFN. (91) 447 34 10 - Y EN LAS MEJORES TIENDAS DE INFORMATICA, TIENDAS Y MAYORISTAS... CUMPLIMENTAMOS SUS PEDIDOS EN 24 HORAS



En este programa nos encontramos ante un modelo histórico, el mítico avión de caza de la II Guerra Mundial.

Nuestra cabina de mandos tiene los siguientes indicadores: horizonte artificial, indicador de velocidad vertical, revoluciones del motor, brújula, altímetro, timón de cola, indicador de giro y deslizamiento e indicador de inclinación.

El conjunto de indicadores y manecillas, reproducen exactamente los controles de este avión, permitiendo un pilotaje preciso y no excesivamente complicado.

Unos minutos de práctica siguiendo los consejos de las instrucciones, nos permitirán hacernos con el aparato; en total manejamos 10 teclas junto con el joystick.

Hasta aquí, podríamos decir que nos encontramos ante un simulador más, brújula, altímetro, tren de aterrizaje, ¿Pero qué aporta de nuevo este programa?

Hemos de decir que lo mejor del Spitfire 40, son sin lugar a dudas los gráficos. En este programa se ha salido del concepto tradicional de simulador de combate, para entrar en una nueva dimensión en esta clase de programas.

La diferencia de los demás, en el Spitfire tenemos la auténtica sensación de encontrarnos en la cabina del piloto.

Tenemos dos vistas básicas; la del tablero de mandos y la de visión del exterior, cada una de las cuales está tratada con una riqueza gráfica, que parecen de verdad.

En el tablero de mandos, los indicadores están compuestos por relojes, manecillas y brújulas, todas hechas reproduciendo exactamente los mandos del avión de la II Guerra Mundial.

La visión a través de la cabina está enmarcada por las barras que soportan el fuselaje, dibujadas con un gran efectismo tridimensional, que nos incita a descubrir lo que hay detrás.

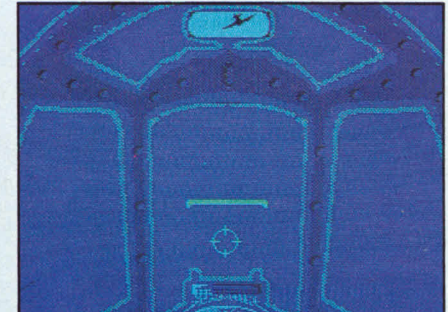
Todos los gráficos están hechos en el modo de dieciséis colores, lo que ayuda a darle espectacularidad al juego.

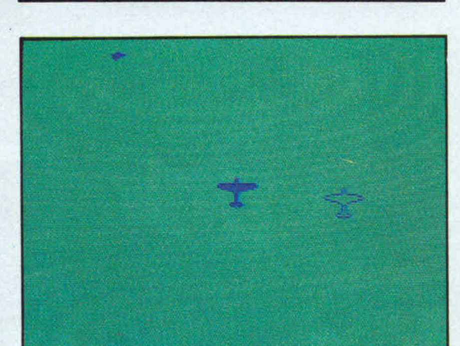
Si hasta aquí el programa parece bueno, no es nada con lo que nos espera en el combate.

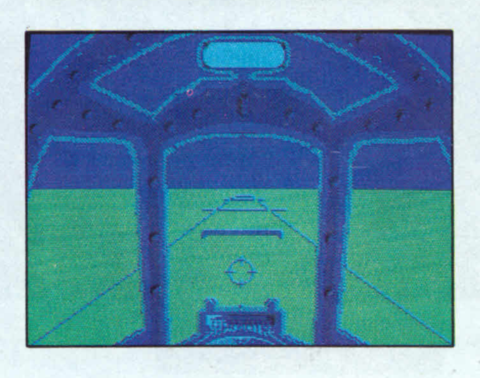
Cuando nos aproximamos al avión enemigo, entonces sí podemos apreciar que es un programa en tres dimensiones: delante de nosotros la negra silueta de un caza, se mueve intentando esquivarnos. Comenzamos a ametrallarle sin piedad, y nuestro adversario intenta un giro para escapar. Cuando nos acercamos a él su tamaño crece desmesuradamente y una certera ráfaga le hace caer.

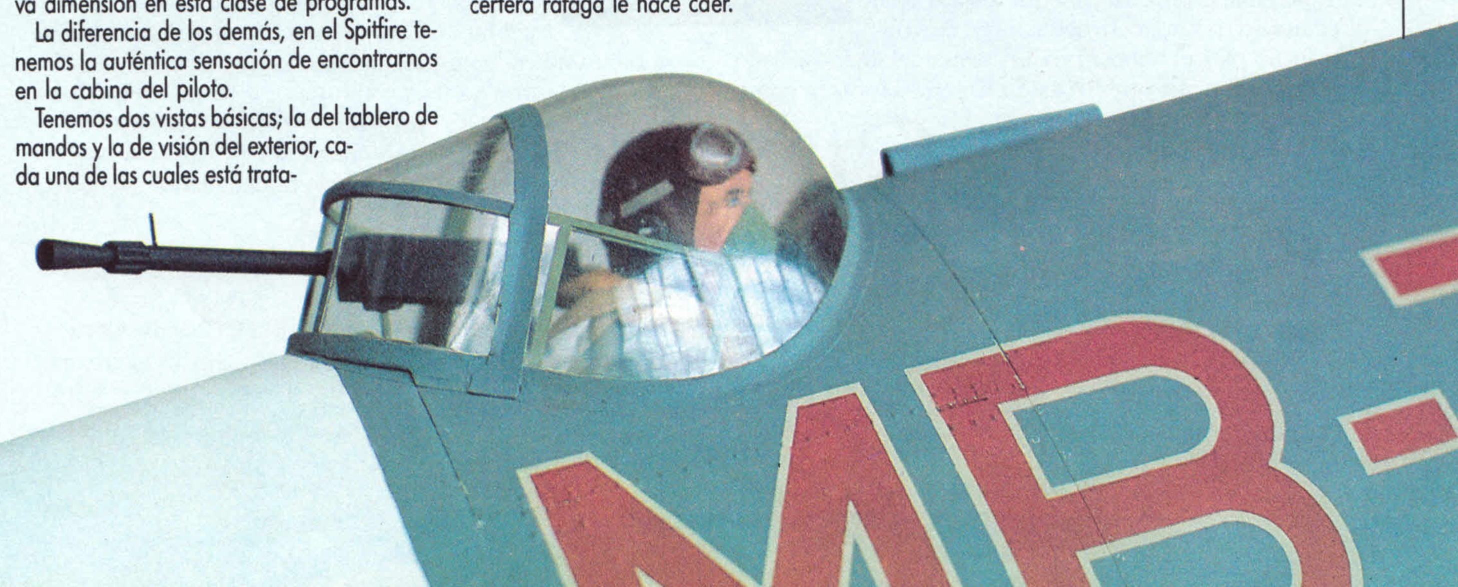
En Spitfire 40 encontramos excelentes gráficos, emoción y tensión; el manejo del aparato no requiere horas de práctica, y la localización de enemigos se realiza de forma bastante rápida. Un programa en el que en la fase de combate se pueden ejecutar técnicas de evasión, picados, loopings y lo que nos apetezca para ponernos a la cola del caza enemigo y aniquilarle.

La casa autora del softwares es Mirrorsoft.









El caza de los Vencedores, Skyfox de DRO SOFT

El número 1 ahora en disco para AMSTRAD

Simulación de vuelo tridimensional de combate aéreo y ataque al suelo.

Cinco niveles, quince escenarios y capacidad de juego estratégico.

Estás en la cabina del caza que sería el sueño de cualquier piloto, pero desde luego eres un mal sueño para el pobre tipo que tienes delante, confiado en una misión sin problemas. Caliéntale la tobera con tus láser y apartate mientras estalla en una bola de fuego. Rápidamente ponte en picado para caer sobre los blindados enemigos, como la peste entre los cerdos.

SKYFOX es el juego que más rápidamente se está vendiendo en toda la historia de Electronic ARTS.

Posee la más asombrosa animación de alta velocidad que hayas visto en tu ordenador.

Ahora puede ser tuyo totalmente traducido al castellano.



EN CASTELLANO

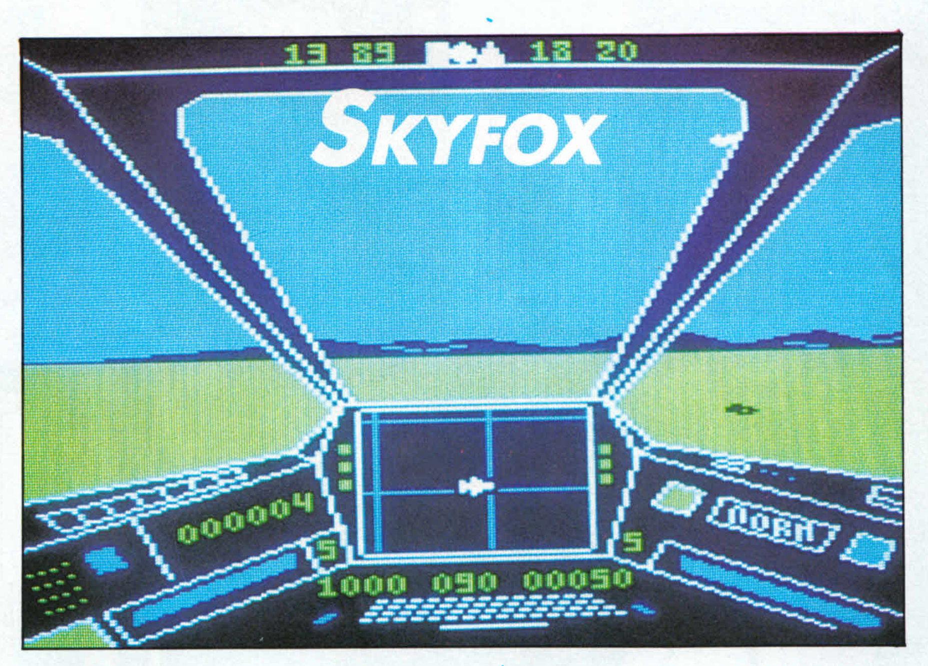
DRO SOFT

P.V.P.: 2500 pts.



CARACTERISTICAS: NACIONALIDAD: Federación galáctica. FABRICANTE: TOBEY ASTRONAUTICS TIPO: Caza interceptor multipropósito PROPULSION AUXILIAR: Un generador antigravitatorio a 66 MKI TRIPULACION: Un humano. ARMAMENTO: Dos cañones láser de fuego continuo de 70 kilojulios 10 toneladas de empuje. 5 misiles rastreadores de calor tipo PHOENIX 5 misiles guiados por radar tipo TYPHOON DEFENSA: — 2 unidades deflectoras WCRC AYUDAS ELECTRONICAS: Radar SCANNER de largo y corto alcance conectable al piloto automático. VELOCIDAD EN ATMOSFERA: — 3.000 MPH (Mach IV a 35.000 piés).

Editado por DRO SOFT. Fundadores, 3 - 28028 MADRID
Tlfs.: 255 45 00/09



Si hasta ahora hemos tratado simuladores de vuelo, en los que el control del aparato requería el manejo de una larga serie de teclas y controles, en Skyfox se simplifica este proceso para hacer que la conducción del caza se realice única y exclusivamente con el joystick.

Con Skyfox, salimos de los típicos simuladores de vuelo, y nos adentramos en una versión más cercana a los arcades. En este programa la acción se desarrolla a ritmo trepidante, y nos vemos envueltos en una lluvia de fuego, aniquilando uno tras otro tanques y aviones enemigos.

Nuestro panel de mandos está compuesto por los siguientes indicadores:

Coordenadas de vuelo, reloj digital, pantalla de radar, brújula digital, indicadores de velocidad y altitud, nivel de combustible, estado del escudo protector y número de misiles disponibles.

En la misma pantalla aparece la cabina del piloto junto con el panel de mando. La visión a través de la cabina es uno de los puntos fuertes de este programa.

En él encontramos los mejores efectos tridimensionales producidos por cualquier programa de esta clase. La sensación de velocidad y acercamiento a los elementos hostiles, es verdaderamente apabullante.

Los gráficos de tanques y aviones aumentan de tamaño considerablemente al acercarnos a ellos, siendo sin duda los más grandes que se ha visto en esta clase de programas.

Esto refuerza considerablemente la sensación de acercamiento y velocidad, pareciendo incluso que podríamos hasta chocar contra ellos.

Una innovación interesante, que facilita considerablemente la aproximación a los cazas enemigos, es la inclusión del piloto automático de intercepción; con este sofisticado artilugio, el ordenador de vuelo del Skyfox nos lleva directamente hacia los aviones detectados por el radar.

De esta forma evitamos largas navegaciones de aproximación a los objetivos, haciendo que el combate se desarrolle de forma rápida y sin que dejemos de accionar el disparador del joystick ni un solo instante.

Para hacer fuego contra blancos difíciles disponemos de dos tipos de misiles: los guiados por el calor y los autodirigidos, con los cuales podemos atacar a las escuadrillas de cazas que aparecen ante nosotros.

Nos encontramos ante el simulador de combate que se maneja con más facilidad. En Skyfox no hay que realizar complicadas operaciones de despegue y mantenimiento del rumbo, etc.

Aquí todo se desarrolla de forma rápida y persiguiendo la total aniquilación de lo que aparezca por pantalla, los blancos son detectados por el radar, y el piloto automático nos dirige a su posición con sólo pulsar un botón.

Un caza de la tercera guerra mundial, que pilotaremos con total soltura, sin haber transcurrido más de dos minutos desde que nos hallamos sentados a los mandos.

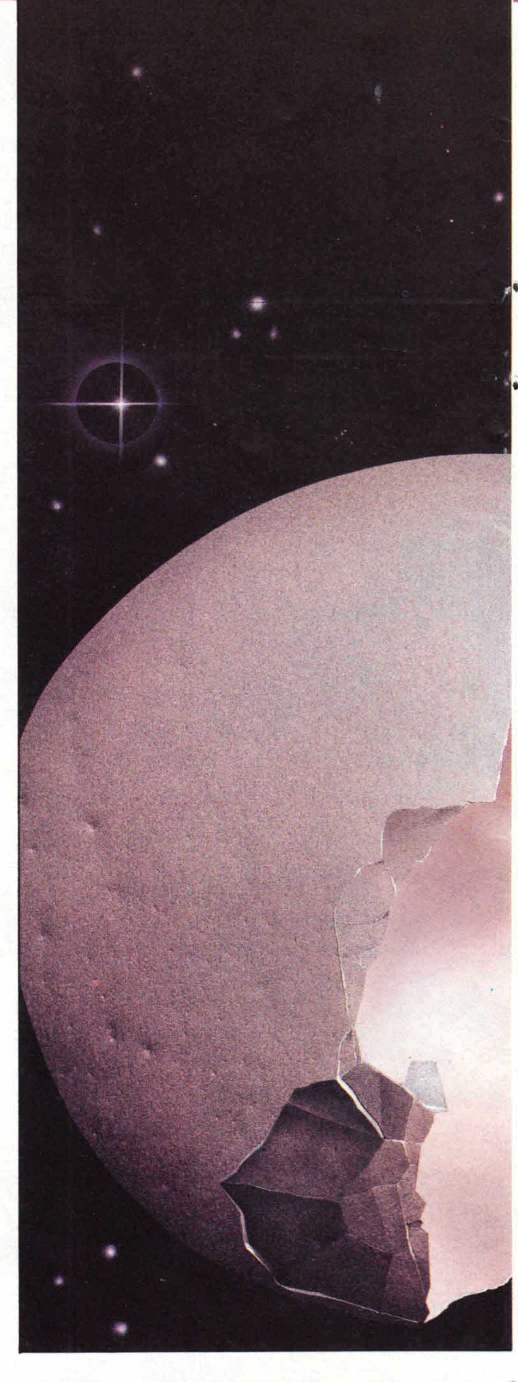
Skyfox es un producto de Electronics Arts.



STREET HAWK
ocean
DRAGON'S LAIR
KIT FROM DATA EAST!
$9.95
SILENT SERVICE
GALIVAN
Imagine
the name of the game
THE GOONIES
COMMODORE 64/128
TRAILBLAZER
MSX 64K
Sydney
B.C. II GROG'S REVENGE
SPECTRUM 48K
TENNIS!
Konami
the name of the game
SPECTRUM
Las Tres Luces de Glaurung
FIST II
MELBOURNE HOUSE
The Home of the Hits!
FOOTBALLER OF THE YEAR
THE GREAT ESCAPE
ocean
ERBE SOFTWARE.
C/. STA. ENGRACIA, 17
DISTRIBUIDOR EXCLUSIVO
PARA ESPAÑA:
28010 MADRID. TEL. (91) 447 34 10
DELEGACION BARCELONA:
AVDA. MISTRAL, N.º 10
TEL. (93) 432 07 31
ERBE
ERBE

ROBOTICA

Los avances técnicos logrados durante los últimos años del siglo XX superan a todos los realizados durante el resto de la Historia del hombre. Si esto ya de por sí es suficiente atractivo, hay un tema cuyo desarrollo lo hace aún mucho más excitante: ¡Estamos hablando del robot! Y ¿qué poder tiene el robot que no tienen otros descubrimientos, incluido el ordenador (mal llamdo «cerbro electrónico»)?



Desde los tiempos más remotos se ha convertido al robot en un mito universal que aparece en todas las culturas y religiones que tienen un mínimo de tradición. Vamos a tomar dos ejemplos representativos de todos los que existen en la nuestra para no extendernos mucho y además porque es la que mejor conocemos. *¿De acuerdo?*.

¿Quién no recuerda la odisea de aquel escultor, llamado Pygmalión, que se enamoró tan tiernamente de una de sus figuras de mármol que conmovió el corazón de la diosa Atenea hasta el punto que dio vida a la obra y la convirtió en una Galatea de carne y hueso?

Y, por cierto, esta historia tiene un reflejo mucho más moderno. Suponemos que a nadie se le habrá pasado por alto el paralelismo que tiene con la historia de Pinocho que más tarde escribió Collodi. Esto nos confirma que a pesar del paso de los años, la idea de la escultura animada sigue interesando.

Pero no son los únicos. La tradición hebrea nos habla de una figura de barro, a la que se da forma bajo las leyes de la Cábala y debe modelarse en un momento especialmente difícil para la Humanidad, para después insuflarle la vida por medio de una Estrella de David colocada a la altura del **«corazón»**. Se trata del **«Golem»**.

La reflexión más importante que podemos sacar de esta historia es que el hombre creaba al Golem para salvar a la Humanidad de algún mal y, una vez superado el peligro, se le destruía. Era como una especie de héroe-esclavo que**«nacía»** con un determinada finalidad y cuando ya había conseguido cumplirla, sencillamente desaparecía.

Además de esta vertiente mitológica, los robots nos atraen por un montón de cosas más. No cabe duda que para la mayoría de las personas es muy difícil dejar atrás esa especie de **«morbo»** ligado íntimamente a los muñecos, las figuras de cera, los androides que representan algo que se queda a medio camino entre la vida y la falta de ella, la imitan pero son incapaces de participar, al menos conscientemente, en alguno de sus aspectos.

Hasta ahora hemos estado utilizando con toda soltura y sin el mínimo recato la palabra **«robot»**, y tenemos que decir que hemos tenido un tremendo lapsus. En el año 1917 se emplea por primera vez, y con propiedad dicha palabra en la obra Opilec de Karen Capek, donde se da forma a cada uno de los conceptos que la componen.

Y es en 1942 cuando se comienza a oír hablar de la **«robótica»** como ciencia que trata de los robots, por supuesto. Isaac Asimov fue el iniciador e impulsor de esta nueva parcela del conocimiento, así como de otras muchas ya conocidas por todos nosotros.

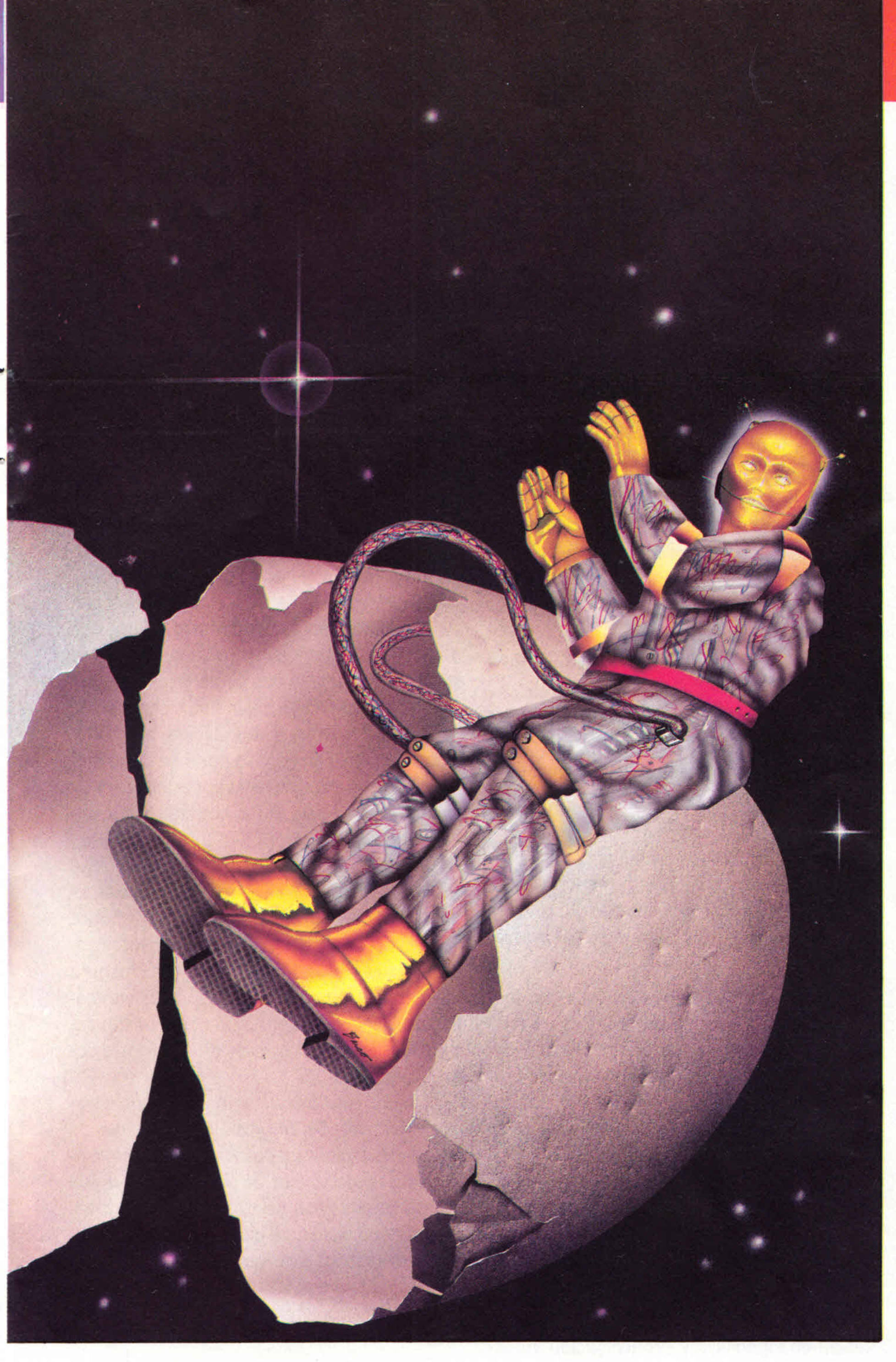
De este modo lo anterior a estas fechas entra dentro del hombre artificial o del autómata, que sería el más claro precedente del robot.

Para conocer datos certeros sobre el campo de los autómatas nos encontramos con problemas verdaderamente insalvables. Muchos de ellos han desaparecido y no podemos comprobar que fueran capaces de realizar las maravillas que sus creadores cuentan largamente en sus crónicas.

Sin embargo, sí tenemos pruebas de verdaderas e ingeniosas maravillas que funcionan a base de aire, vapor de agua o complicados sistemas de relojería que, como ya hemos dicho, pueden considerarse antecesores *(lejanos, eso sí)* de los robots. Pero dejemos este tema aconsejando a los interesados que busquen en su localidad algún museo donde estén recogidas algunas de estas sorprendentes máquinas.

Fue a partir de la Segunda Guerra Mundial cuando las grandes empresas comenzaron a interesarse por la automatización para mejorar la producción en cadena de fábricas y talleres. Los primeros robots industriales se utilizaron en aplicaciones tales como coger piezas y colocarlas en un determinado sitio.

Dentro de este campo hay que destacar la figura de George C. Deval, considerado como el padre de la robótica. Desde nuestra perspectiva actual sus creaciones pueden parecernos un poco ingenuas. Por otra parte era un fiel seguidor de Asimov, lo que quiere decir que quizá fuera más imaginativo que teórico, pero en 1954 ya había postulado todos los elementos y conceptos que forman parte de los actuales robots industriales. ¿Está de acuerdo con nosotros en que todo eso le convierte en el verdadero padre de la **«robótica»**?



Sin ir más lejos reconoció que todavía existía un largo camino por recorrer en este campo ya que los prototipos de su creación tenían grandes deficiencias. Y también contagió su entusiasmo a Joseph Engelberger (otro enamorado de Asimov), creador de la primera empresa cuya finalidad era la fabricación robótica. Hoy en día la Engelberger Unimation Inc. continúa siendo una de las mayores suministradoras de robots industriales.

Para ser sinceros donde la robótica ha experimentado un verdadero auge tanto por el interés que despierta su estudio como por su utilización masiva ha sido, sigue siendo y probablemente lo será por mucho tiempo en Japón. En este país la robótica industrial no sólo dobla las cantidades de cualquier país industrializado, sino que se renueva constantemente. Día a día podríamos decir sin miedo a exagerar.

Hasta aquí puede servir la introducción, pero entremos directamente en el tema. Y así, de golpe, lo primero que se nos ocurre es preguntarnos: *¿qué es un robot?*

Todos nosotros tenemos dos ideas claramente diferenciadas, que son incluso contradictorias, respecto a la palabra robot. Por un lado tenemos una tradición cultural, de la que ya hemos hablado, potenciada poco a poco por la literatura de ficción *(el género de ciencia-ficción, para ser exactos)* y más recientemente el cine.

Según esta idea romántica, el robot es un ser hecho a semejanza del hombre (más o menos), con unas capacidades cada vez más humanas, de forma que algunos son capaces de tomar decisiones por sí mismos y otros tienen claros deseos de inmortalidad o incluso de procreación.

Frente a este concepto se encuentra la realidad. No todos los robots están creados a imagen del hombre, es más, la mayor parte de ellos están fabricados imitando funciones humanas, no su aspecto. Se les diseña funcionalmente para que sean capaces de realizar una serie de trabajos muy definidos tal como decíamos anteriormente al referirnos a los robots industriales.

Bueno pero, **¿qué es un robot?**. Tal vez podamos decir que es una máquina con capacidad para ser programada y que, por tanto, puede memorizar una secuencia de operaciones y repetirla incansablemente.

Parece un definición muy pobre pero, sin embargo, es lo suficientemente amplia como para abarcar las diferentes generaciones de robots así como sus distintas configuraciones mecánicas y funcionales.

Tal vez debíamos olvidar por un momento la pregunta anterior, realizada tan directamente, para intentar rodearla y llegar a encontrar su respuesta a través del estudio de la robótica actual y así desde el conocimiento de casos particulares tal vez seamos capaces de llegar a un concepto general válido.

De momento, lo que sí parece claro, y aplicable a la mayoría de los casos, es que el robot es una máquina especializada. Y cada uno está pensado, diseñado y montado para cumplir su misión con la mayor exactitud.

Vamos a verlos despacito y de cerca. Para ello los dividiremos en dos grandes apartados: los que no se parecen en nada a los humanos, en cuanto a su estructura y aspecto, y los llamados «androides».

Y antes de empezar a recorrerlos uno por uno, vamos a pedir disculpas a los robots de la lavadora, el horno, la plancha, etc. porque en una sabia decisión unilateral hemos decidido sacarles de la definición de robot. Para ser más exactos diremos que están mucho más cerca de ser pequeños ordenadores. En fin, perdón.

Son ya muy antiguos los mecanismos que tienen por finalidad la manipulación a distancia. Pero es a partir de la Segunda Guerra Mundial y la necesidad de trabajar con material radiactivo la que impulsa el perfeccionismo de estas manos teledirigidas. Desde este momento puede hablarse de una verdadera especialización de estas máquinas y de los brazos de robots, que son los que nos interesan en este caso y que pueden considerarse sus descendientes.

Los fines para los que se crearon fueron meramente industriales. Se usan para coger, colocar, atornillar, perforar, etc. Como verá se trata de realizar trabajos muy específicos dentro de una cadena de montaje, sustituyendo la operación mecánica del hombre. Tienen la ventaja de que no se cansan, no se distraen y tienen una mayor fuerza, pero en cambio son mucho más pesados.

Pero lo verdaderamente interesante de estos brazos, aunque no es una de las áreas más explotadas, es su combinación con un vehículo

móvil. Ciertamente este conjunto perdería algo de fuerza en comparación con un brazo sólidamente apoyado, pero a cambio facilitaría su presencia allí donde fuera necesario.

Si está unido a un ordenador podríamos conseguir evitar choques. Dirigido desde una pantalla, por ejemplo, el robot móvil recoge y transporta cajas de un almacén, libros de una biblioteca, etc. para después colocarlos con precisión en su lugar exacto.

En cuanto a las bases sobre las que están colocados estos brazos son también robots. Están unidas a un ordenador que las dirige a través de unas **«pistas»** que están formadas por raíles, cables eléctricos e incluso líneas en color. Es evidente que en este último caso deberán estar provistas de unas células sensibles a la luz.

Todos ellos deben disponer de sensores que eviten choques en cualquier circunstancia, deteniendo el vehículo cuando en su camino aparezcan obstáculos más o menos imprevistos, incluido el operario humano que tiene todo el derecho del mundo a cruzar la fábrica o el almacén.

Pero no sólo podemos destinarlos a estas misiones tan concretas. Sofisticando y complementando estos brazos con una serie de células, sensores, luces, cámaras de T.V., etc. son capaces de realizar operaciones mucho más delicadas en lugares donde, bien por condiciones hostiles, bien por economía o por ambas cosas a la vez, sea más beneficiosa la presencia del robot que la humana.

Hablamos, por ejemplo, del espacio exterior, donde la toma de muestras se realiza con estos brazos aplicados a las naves, satélites o vehículos espaciales. Y podemos aplicar esto mismo al fondo del mar, las zonas subterráneas, las prospecciones petrolífera; también pueden resultarnos imprescindibles en algo realmente peligroso: las minas. En este caso estarán dotados también de algún accesorio especial que corte las rocas antes que el brazo comience a actuar.

Otra aplicación importante de este tipo de herramientas es puramente militar. Son unos desactivadores de explosivos realmente eficaces y no ya sólo por la importantísima circunstancia de salvar vidas humanas, sino porque ellos mismos tienen lo que nosotros entenderíamos como «instinto de conservación» y se autoprotegen de la mejor forma para que los daños que sufran sean menores, con lo que la parte económica sale también beneficiada.

Pero a nosotros, usuarios, entusiasmados y enloquecidos con nuestro **Amstrad,** lo que nos interesa más bien es la aplicación que uno de estos brazos podría tener respecto a nuestro ordenador.

En efecto, desde hace ya unos años algunas casas han comercializado unos genios cuya finalidad esencial es la educativa. Un brazo de robot en casa va a servirnos ni más ni menos que para aprender su funcionamiento, sus posibilidades, su manejo. Para investigar por nosotros mismos e intentar crear o mejorar un programa que lo mueva con precisión. Un ejemplo clásico que le brindamos es intentar jugar al ajedrez o las damas con un brazo educacional y un buen programa, quizá codificado por nosotros mismos, ejecutado por nuestro ordenador.

La gran diferencia entre estos brazos y los industriales es el tamaño y, consecuentemente la fuerza que desarrollan. Por tanto, la gran posibilidad que tenemos es intentar potenciar sus habilidades, por ejemplo, cambiando la pinza (o dedos robóticos) por imanes o viceversa. Incluso los más **«manitas»** pueden intentar realizar sus propias creaciones con diferentes materiales, darle más potencia (y quizá nos ayude a colocar nuestros libros, discos o vídeos). No se limite a montar los kits tal como vienen preparados de la fábrica, sino intente hacer un robot a su medida.

Pero quizá mucho más divertidos que estos brazos sean los robots de suelo. Su fabricación no tiene por qué ser más cara que la de los anteriores y, sin embargo, sus posibilidades son bastante superiores.

Su diseño abarca desde la imitación de las más avanzadas tecnologías hasta los clásicos animales de peluche que son, sin duda, de lo más atractivo para los «informáticos» más jovencitos.

Este tipo de robots tiene además una mayor autonomía que los brazos. Por un lado pueden unirse al ordenador por medio de un cable, pero también se pueden utilizar rayos infrarrojos para dirigirlos, aunque en este caso necesitaremos un espacio diáfano entre uno y otro.

En cuanto a sus posibilidades podemos decir que son infinitas. Son capaces de realizar increíbles dibujos y adaptándoles células y sensores que le ayuden a orientarse, evitar obstáculos o cualquier otra cosa que se nos ocurra. Podemos desarrollar toda nuestra experiencia en ingeniería y programación y conseguir adaptar a estos robots de suelo cualquiera de las novedades que podamos imaginar.

Y pasamos ya a los que por su aspecto y funcionamiento son más semejantes a los humanos: los androides. Son, quizá, los más sugerentes pero no por ello los más avanzados técnicamente. Antes de entrar de lleno en ellos, vamos a hacer una advertencia, quizá un tanto superflua, pero que es necesaria para impedir que nuestra imaginación se dispare.

Al hablar de este tipo de robots todos nuestros recuerdos y experiencias se centran en los ejemplos que nos ha mostrado el cine. Todos conocemos a Roddy, a C3PO y R2D2 y alguna que otra estrella del momento. Pero todos ellos son productos de la imaginación literaria, la ciencia no va por ese camino. De hecho tendrán que pasar bastantes años para conseguir prototipos semejantes.

Y además parece que su aspecto físico no interesa en demasía a los científicos que dan más importancia a la especialización y funcionalidad.

Dicho esto, continuemos. Los menos sofisticados son los llamados **«maniquíes»**, una serie que intenta imitar la apariencia humana o animal con la mayor perfección posible. Sus movimientos son toscos, limitados y su capacidad de memoria mínima. Son típicos en los vídeos, escaparates y en alguna que otra manifestación a modo de reclamo. Por todo ello el interés que tienen para nosotros es también bastante anecdótico y superfluo.

¿Recuerda la película «Almas de metal»? Es algo que, aunque no sea de vital importancia, al menos puede resultar muy divertido. Se trata de una serie de robots con una perfecta apariencia de estar dotados de vida que les hacía vivir su aventura favorita a cualquier humano que pudiera costeársela.

¿Puede imaginarse a sí mismo luchando con un **«marcianito»** de tres dimensiones en lugar de enfrentársele en la pantalla de su monitor? Con un poco de tiempo, paciencia y perfecionamiento podremos conseguir hacerlo realidad.

Una aplicación mucho más interesante de los *«androides»* es la que consiste en **«rellenerles»** de microprocesadores, conectados a su vez con uno central, que imiten los órganos vitales de un humano. De esta manera pueden experimentarse situaciones límite reales, sin el menor peligro, que nos permitan desarrollar métodos capaces de evitarlos o corregirlos.

Los más conocidos son los llamados **«domi»**. Se trata de un grupo de androides de diferentes tamaños y pesos que trabajan dentro de los automóviles. Se puede decir que son los **«inventores»** del cinturón de seguridad, cabezales, colchón de aire... Nos explicamos, los domis son los encargados de demostrar las consecuencias que los accidentes de automóvil pueden acarrearle al ser humano de verdad.

Acompañándoles están sus primos hermanos, que trabajan en la Facultad. Si los domis han sufrido toda serie de traumas imaginables, sus familiares se han visto afectados por paros cardíacos, asfixias, tumoraciones y cuantos etcéteras se le puedan ocurrir.

Después de quitarnos el sombrero ante ellos, seamos prácticos. ¿Qué pasa con los androides y los ordenadores caseros?

Pues poco más o menos lo mismo que ocurriría en el caso de los brazos de robot. Por ahora estamos mucho más cerca del juego educativo que de la utilidad realmente práctica.

Dentro de los que tenemos en casa, podemos hacer dos grandes grupos: los de juguete y los domésticos. La verdad es que su apariencia no difiere mucho ya que el cine influye en gran manera sobre los gustos de las personas y R2D2 ha tenido demasiado **«gancho»**.



Los androides de juguete se han desarrollado sobre todo en Japón, cómo no. Su apariencia es cada día más benigna y amistosa, aunque todavía recuerdan en parte sus orígenes belicosos, cuando iban cargados de armas espaciales, pistolas de rayos láser o de cualquier otro tipo (simuladas claro).

Sus limitaciones vienen dadas por las del microprocesador que llevan incorporado y que permite programarles para que ejecuten y recuerden una serie de movimientos e incluso alguna que otra frase que podrán intercambiar con los humanos, una vez reconozcan su voz.

En cuanto a los robots domésticos se puede decir que todos ellos se mueven sobre sí mismas, o sea, son autónomas. Se controlan por un ordenador que generalmente llevan en su interior, o bien mediante infrarrojos que lo unen con un micro de mesa.

Pero realmente, *¿qué es lo que pueden hacer?* Aparentemente multitud de tareas caseras pueden ser realizadas por estos androides. Parece que se acabó el barrer, poner la mesa, ordenar un armario. La situación, sin embargo, es muy diferente. La tecnología con la que están fabricados no está lo sufientemente desarrollada como para que sean unos perfectos ayudantes del ama de casa. De momento nos conformaremos con aprender robótica, electrónica y mecánica con ellos, así como idear un programa que podamos **«meter»** en su memoria y conseguir perfeccionar o incluso ampliar las primitivas tareas de las que están dotados.

Pero lo más atractivo de estos robots es que, al igual que ocurría con los brazos, con un prototipo, nuestro ordenador y nuestra imaginación, podemos aprender, investigar, inventar y desarrollar cualquier idea por peregrina que nos parezca. Tal vez al principio los resultados no sean todo lo impresionantes que deseemos pero de lo que no cabe duda alguna es que nuestros conocimientos aumentarán. ¿Por qué no se anima y fabrica un prototipo?

A estas alturas no sabemos si estamos en condiciones de responder a aquella pregunta que hicimos sobre la naturaleza de los robots, pero a cambio haremos una reflexión **«filosófica»**.

Hay quien ve en los robots una amenaza. Bueno, pues no lo son.

En principio las posibilidades del robot son todavía limitadas y no pueden considerarse en ningún momento como peligrosas.

Sin embargo, sí podemos tener en ellos una gran ayuda ya que se «encargan» de todas las tareas pesadas y rutinarias, dejando a los humanos las labores creativas y, por supuesto, una mayor cantidad de tiempo libre.

En un futuro el robot cumplirá una labor semejante a la del antiguo esclavo, pero evitando el problema moral que la esclavitud significa en este caso.

El empleo de este concepto es de muy dudosa aplicación al referirnos al trabajo que realiza un robot.

Si le parece que esta premisa es lo suficientemente lógica piense por un momento que hemos llegado al futuro.

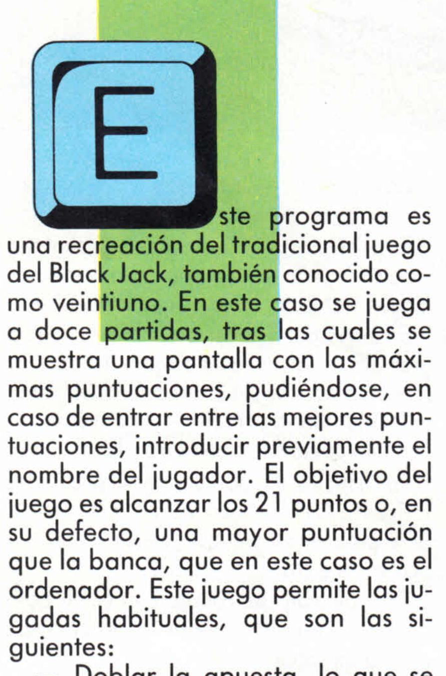
Los robots serán lo suficientemente inteligentes como para descubrir que el papel de los humanos es más apasionante que el suyo.

Y entonces...

BLACK JACK

Los juegos de cartas por ordenador, al parecer, constituyen un tema favorito de los programadores y nuestros lectores no son una excepción. Una y otra vez llegan a nuestra redacción multitud de ellos, unos muy buenos, otros menos buenos, pero todos bienvenidos.
Hemos seleccionado uno de ellos, que «computariza» el famoso juego inglés del «Black Jack», conocido en España como 21 y, pariente cercano de las «siete y media».
El autor del programa ya se toma la molestia, unas líneas más abajo, de describir el juego con detalle, así que no lo repetiré aquí. Sólo quisiera llamar la atención de los lectores acerca de la precisión con que están realizados los gráficos de las caras, así como la original y, acertada, disposición de las mismas en la pantalla.
Más de una vez hemos dicho en la revista que, por la magia del ingenio, el byte se convierte en padre y madre del Arte. Black Jack es una prueba de ello.

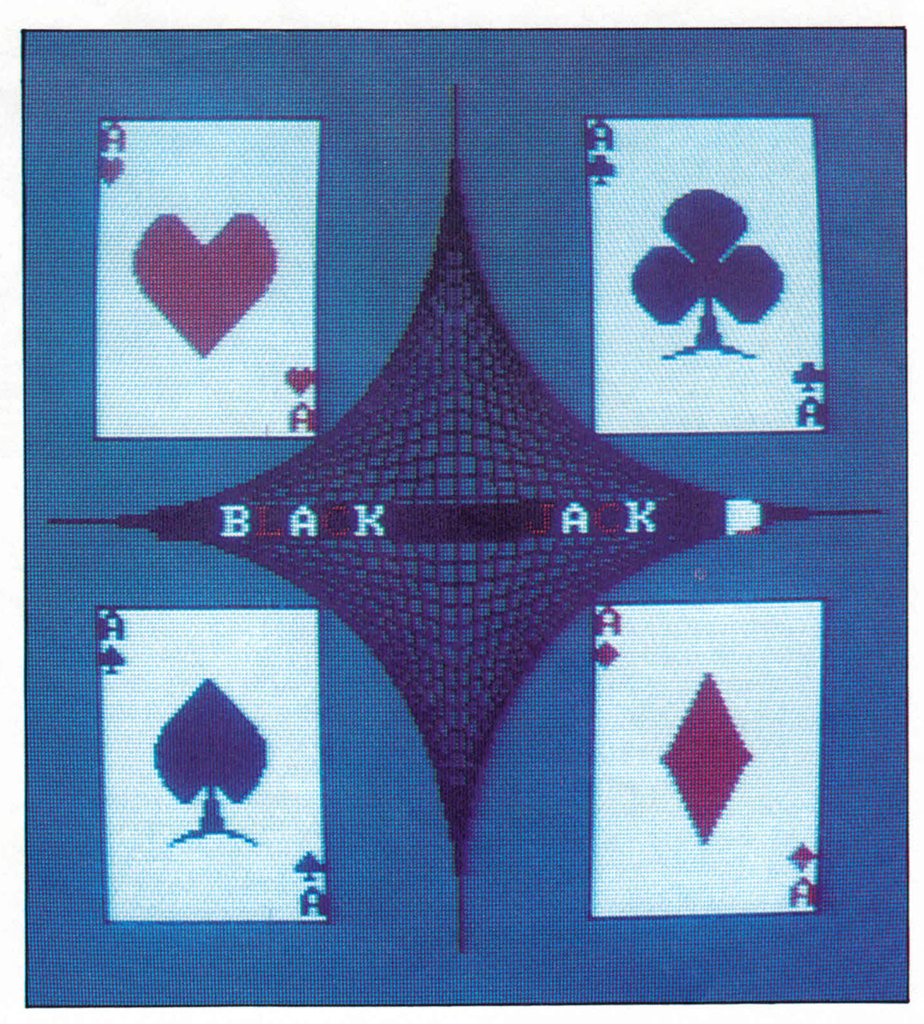
Antonio Fernando Aguilera Romera



Este programa es una recreación del tradicional juego del Black Jack, también conocido como veintiuno. En este caso se juega a doce partidas, tras las cuales se muestra una pantalla con las máximas puntuaciones, pudiéndose, en caso de entrar entre las mejores puntuaciones, introducir previamente el nombre del jugador. El objetivo del juego es alcanzar los 21 puntos o, en su defecto, una mayor puntuación que la banca, que en este caso es el ordenador. Este juego permite las jugadas habituales, que son las siguientes:

— Doblar la apuesta, lo que se puede hacer cuando nuestra puntuación es de 9, 10 ó 11 puntos.

— Asegurarse, es decir, apostar la mitad de lo que se lleva apostado a la banca. Si ésta obtiene un black Jack (un as y una carta de valor diez), el jugador gana esa cantidad.



— Abrir el juego en dos ramas independientes. Esto sólo se puede hacer cuando las 2 primeras cartas del jugador son del mismo valor y éste no ha doblado la apuesta. Existe una jugada que supera a todas las demás: ésta es el Black Jack, que se obtiene cuando se tiene un as y una carta de valor diez (10, J, Q, K). Caso de no alcanzarse ni Black Jack ni 21 por ninguna de las 2 partes, ganará el que más se acerque a 21.

ESTRUCTURA

10-240	Inicialización
250-310	Presentación
350-1440	Bucle principal
360-650	Baraja y pide apuesta
660-1110	Juega el jugador
1120-1370	Juega el ordenador
1380-1440	Fin del bucle principal
1450-1470	Pregunta si se vuelve a jugar
1480-1910	Resultado de la partida
1920-3170	Dibuja la carta
3180-3270	Dibuja el marco y el fondo de la carta
3280-3330	Identificador del naipe
3340-3390	Imprime el mensaje
3400-3460	Extrae una carta del mazo
3470-3520	Espera que se expulse una tecla
3530-3580	Subrutinas auxiliares
3590-3940	Máximas puntuaciones
3950-4040	Escribe ventana de puntuación

VARIABLES

a$	Cadena con los números de las cartas
b$	Cadena con los palos de las cartas
rec (), recS()	Matriz con la puntuación y nombre (máximas puntuaciones)
e$	Cadena con la baraja
s	Puntuación del 1.° camino
p	Puntuación del 2.° camino
t	Puntos de la banca
bj1, bj2	Indicadores del Black Jack en los respectivos caminos
dg	Dinero acumulado
d,e	Coordenadas de impresión de la carta
q	Cantidad del seguro
h	apuesta
fl	Bandera de fin de juego del jugador
r	Bandera de apuesta doblada
v	Valor de la 1.ª carta del ordenador
u	Valor de la 1.ª carta del jugador
bdna	Bandera de dinero acabado
k	Dinero ganado (o perdido) en una partida
m	Dinero ganado en todas las partidas jugadas
a	Carta del mazo
c	Palo de la carta

Nota.—*Para ver correctamente el listado, ejecutar en modo directo: SYMBOL AFTER O: LIST.*



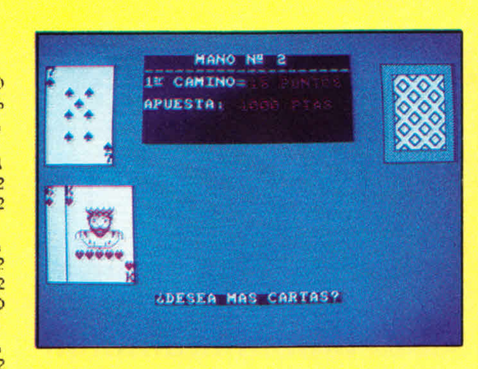

```
10 REM ***
****************************
20 REM *           BLACK JACK
 *
30 REM ***************************
**
40 REM *       Antonio Fernando
 *
50 REM *       Aguilera Romera
 *
60 REM *            GRANADA
 *
70 REM ***************************
**
80 SYMBOL AFTER 0
90 MODE 1
100 REM ***************************
***
110 REM *         INICIALIZACION
  *
120 REM ***************************
***
130 INK 0,10:INK 1,26:INK 2,0:INK 3
,6
140 BORDER 10:PAPER 0:PEN 1:CLS
150 SYMBOL 163,39,101,37,37,37,37,1
19,0
160 SYMBOL 255,235,236,136,232,0,25
4,0,0
170 SYMBOL 254,248,216,216,248,0,24
8,0,0
180 SYMBOL 253,0,0,127,0,0,127,0,0
190 SYMBOL 252,192,224,176,152,152,
176,224,192
200 a$="A23456789#JQK":b$=CHR$(228)
+CHR$(227)+CHR$(226)+CHR$(229)
210 RANDOMIZE TIME:c$=STRING$(40,32
)
220 f$=CHR$(207)+CHR$(200)+CHR$(201
)+CHR$(200)+CHR$(201)+CHR$(200)+CHR
$(207):f1$=CHR$(207)+CHR$(201)+CHR$
(200)+CHR$(201)+CHR$(200)+CHR$(201)
+CHR$(207)
230 e$="":FOR i=1 TO 52:e$=e$+CHR$(
i):NEXT i
240 DIM rec(6),rec$(6):FOR i=1 TO 6
:rec(i)=80000+(60000-(i+1)*5000):re
c$(i)="AMSTRAD":NEXT
250 REM ***************************
*
260 REM * PANTALLA DE PRESENTACION
*
270 REM ***************************
*
280 ORIGIN 320,200:FOR px=0 TO 200
STEP 10:PLOT px,0,2:DRAW 0,200-px:D
RAW -px,0:DRAW 0,-(200-px):DRAW px,
0:NEXT:ORIGIN 0,0
290 a=1:d=1:e=10:b=3:c=1:GOSUB 1930
:SOUND 1,INT(RND*100+100),25:d=15:e
=25:c=2:b=3:GOSUB 1930:SOUND 1,INT(
RND*300+100),25:d=1:b=2:c=3:GOSUB 1
930:SOUND 1,INT(RND*500+100).25:d=1
5:e=10:c=4:b=2:GOSUB 1930:SOUND 1,I
NT(RND*700+100),25
300 x$="  BLACK     JACK  ":FOR I=1
TO 17:PAPER 2:PEN 1+(2 AND I/2=INT(
I/2)):LOCATE I+11,13:PRINT MID$(x$,
i,1);:FOR ret=1 TO 5:NEXT ret:NEXT
i
310 FOR ret=1 TO 2000:NEXT
320 REM ---------------------------
----
330 REM -AQUI COMIENZA BUCLE PRINCI
PAL-
340 REM ---------------------------
----
350 dg=80000:m=0:k=0:FOR mano=1 TO
12
360 REM ***************************
*
370 REM *   BARAJA Y PIDE APUESTA
*
380 REM ***************************
*
390 qp=INT(RND*100+35)
400 MODE 1:PAPER 0:PEN 1:CLS:d$="BA
RAJANDO":GOSUB 3370
410 INK 3,24
420 FOR I=1 TO qp:x=INT(RND*52+1):y
=INT(RND*52+1):x$=MID$(e$,x,1):MID$
(e$,x,1)=MID$(e$,y,1):MID$(e$,y,1)=
x$:SOUND 1,0,1:NEXT i
430 h=0:l=0:n=0:o=0:p=0:q=0:r=0:s=0
:t=0
440 CLS
450 men$="  ENTRE SU APUESTA  ":LOC
ATE 1,12:PAPER 0:PRINT c$:PAPER 2:P
EN 1:LOCATE 11,12:FOR i=1 TO LEN(me
n$):PRINT MID$(men$,i,1);:SOUND 129
,100,4:WHILE SQ(1)>127:WEND:NEXT i:
LOCATE 11,11:PRINT STRING$(20,32):L
OCATE 11,13:PRINT STRING$(20,32)
460 LOCATE 11,10:PRINT STRING$(20,3
2):LOCATE 11,14:PRINT STRING$(20,32
)
470 PLOT 184,200,3:DRAWR 270,0
480 DRAWR 0,30:DRAWR -272,0:DRAWR 0
,-30
490 PLOT 172,188,1:DRAWR 294,0
500 DRAWR 0,54:DRAWR -294,0:DRAWR 0
,-54
510 PLOT 160,176,3:DRAWR 318,0:DRAW
R 0,78:DRAWR -318,0:DRAWR 0,-78
520 IF dg>9999 THEN maxap=10000 ELS
E maxap=dg
530 men2$="ENTRE 200 Y"+STR$(maxap)
+" PTAS:......":men2=LEN(men2$):IF
(men2 MOD 2)=0 THEN men2$=" "+men2$
540 PAPER 0:PEN 1:LOCATE 20-INT(men
2/2),22:FOR i=1 TO LEN(men2$):PRINT
 MID$(men2$,i,1);:SOUND 1,200,4:WHI
LE SQ(1)>127:SOUND 128,100,1:WEND:N
EXT i:PRINT STRING$(6,8);
550 h=0:t$="":x$="":WHILE LEN(x$)<6
 AND t$<>CHR$(13)
560 t$=UPPER$(INKEY$):IF t$="" THEN
 560
570 IF ASC(t$)=13 OR (ASC(t$)>47 AN
D ASC(t$)<58) THEN x$=x$+t$:PRINT t
$;:SOUND 1,20+VAL(t$)*10,3:WHILE SQ
(1)>127:WEND ELSE IF ASC(t$)=127 AN
D LEN(x$)>0 THEN lo=LEN(x$):lo=lo-1
:x$=MID$(x$,1,lo):PRINT CHR$(8);CHR
$(16);"."+CHR$(8); ELSE  GOTO 560
580 WEND
590 h=VAL(X$)
600 men3$="POR FAVOR, ENTRE 200 Y"+
STR$(maxap)+" PTAS":men3=LEN(men3$)
:IF (men3 MOD 2)=1 THEN men3$=men3$
+"."
610 IF h<200 OR h>maxap THEN LOCATE
 1,25:PRINT SPC(39);:LOCATE 21-INT(
LEN(men3$)/2),25:PRINT men3$:FOR I=
1 TO 2000:NEXT I:LOCATE 1,25:PRINT
SPC(39);:LOCATE 1,22:PRINT c$:GOTO
540
620 LOCATE 1,22:PRINT c$:din$="APUE
STA TOTAL:"+RIGHT$(STR$(h),LEN(STR$
(h))-1)+" PTAS":IF (LEN(din$)/2)<>I
NT(LEN(din$)/2) THEN din$=din$+"."
630 LOCATE (20-LEN(din$)/2)+1,22:PE
N 2:PRINT din$
640 PLOT (20-(LEN(din$)/2))*16-8,39
9-22*16-4,1:DRAWR LEN(din$)*16+12,0
:DRAWR 0,28:DRAWR -(LEN(din$)*16+12
),0:DRAWR 0,-28
650 FOR I=1 TO 2000:NEXT I:PAPER 0:
CLS:INK 3,6
660 REM ***************************
*
670 REM *      JUEGA EL JUGADOR
*
680 REM ***************************
*
690 WINDOW #1,11,30,1,8:PAPER #1,2:
FOR i=1 TO 8:LOCATE #1,1,1:PRINT #1
,CHR$(11):STRING$(20,32);:NEXT
```

```
700 rcj1$="":rcj2$="":bj1=0:bj2=0:r
=0:f1=0:WHILE r<>1 AND f1=0
710 n=n+1
720 IF n>2 THEN GOTO 910
730 IF n=1 THEN o=o+1:GOSUB 3430:GO
SUB 3450:t=a:d=1:e=34:b=1:GOSUB 321
0:PAPER 1:PEN 2:LOCATE 34,2:PRINT S
TRING$(7,207):FOR i=3 TO 9 STEP 2:L
OCATE 34,i:PRINT f$:LOCATE 34,i+1:P
RINT f1$:NEXT i:LOCATE 34,10:PRINT
STRING$(7,207):GOSUB 3220:GOSUB 358
0:GOTO 750
740 o=o+1:d=1:e=1:GOSUB 3430:GOSUB
1950:GOSUB 3450:t=t+a:v=a
750 o=o+1:d=12:e=n*2-1:GOSUB 3430:G
OSUB 1950:GOSUB 3450:s=s+a
760 GOSUB 3980
770 IF o=2 THEN u=a
780 IF n=1 THEN 710
790 IF s=11 AND (u=1 OR u=10) THEN
bj1=1:f1=1:GOSUB 3980:GOTO 1110
800 IF v=1 THEN d$=CHR$(174)+"DESEA
 ASEGURARSE?":GOSUB 3370:GOSUB 3500
:PAPER 0:LOCATE 1,23:PRINT C$:IF x$
="S" THEN q=INT(h/2):LOCATE 1,25:PR
INT "SEGURO DE";Q;"PTAS":GOSUB 3980
:FOR I=1 TO 1500:NEXT:LOCATE 1,25:P
RINT c$;
810 IF s>8 AND s<12 THEN d$=CHR$(17
4)+"DESEA DOBLAR LA APUESTA?":GOSUB
 3370:GOSUB 3500:PAPER 0:LOCATE 1,2
3:PRINT c$:IF x$="S" THEN r=1:h=2*h
:LOCATE 1,25:PRINT "APUESTA DOBLADA
 A";h;"PTAS":GOSUB 3980:FOR i=1 TO
1500:NEXT i:LOCATE 1,25:PRINT c$;
820 IF s<>2*u OR r=1 THEN rcj2$="N"
:GOTO 910
830 d$=CHR$(174)+"DESEA ABRIRSE?":G
OSUB 3370:GOSUB 3500:PAPER 0:LOCATE
 1,23:PRINT c$;
840 IF x$="N" THEN rcj2$="N":GOTO 9
10
850 d=12:e=34:GOSUB 3430:GOSUB 1950
:GOSUB 3450:p=a:GOSUB 3980
860 o=o+1:e=3:GOSUB 3430:GOSUB 1950
:GOSUB 3450:s=u+a:GOSUB 3980
870 IF s=11 AND (u=1 OR u=10) THEN
bj1=1
880 o=o+1:e=32:GOSUB 3430:GOSUB 195
0:GOSUB 3450:p=p+a:GOSUB 3980
890 IF p=11 AND (a=1 OR a=10) THEN
bj2=1
900 GOSUB 3980
910 IF s>21 THEN rcj1$="N"
920 IF bj1=1 THEN rcj1$="N"
930 IF rcj1$="N" OR bj1=1 THEN 1000
940 d$=CHR$(174)+"DESEA MAS CARTAS?
":GOSUB 3370:IF p THEN LOCATE 3,12:
PEN 1:PRINT "*":GOSUB 3560
950 GOSUB 3500:LOCATE 3,12:PAPER 0:
PRINT " ":GOSUB 3560
960 rcj1$=x$
970 IF rcj1$="N" THEN GOTO 1000
980 o=o+1:d=12:e=n*2+1:GOSUB 3430:G
OSUB 1950:GOSUB 3450:s=s+a:GOSUB 39
80
990 ''''''''''''
1000 IF p>21 THEN rcj2$="N"
1010 IF bj2=1 THEN rcj2$="N"
1020 IF rcj2$="N" OR bj2=1 THEN 109
0
1030 d$=CHR$(174)+"DESEA MAS CARTAS
?":GOSUB 3370:IF p THEN LOCATE 38,1
2:PEN 1:PAPER 0:PRINT "*":GOSUB 357
0
1040 GOSUB 3500:PAPER 0:LOCATE 38,1
2:PRINT " ":GOSUB 3570
1050 rcj2$=x$
1060 IF rcj2$="N" THEN GOTO 1090
1070 o=o+1:d=12:e=34-n*2:GOSUB 3430
:GOSUB 1950:GOSUB 3450:p=p+a:GOSUB
3980
1080 IF p>21 THEN rcj2$="N"
1090 IF bj1=1 AND bj2=1 THEN f1=1
1100 IF rcj1$="N" AND rcj2$="N" THE
N f1=1
1110 WEND
1120 REM ****************************
*
1130 REM *   JUEGA EL ORDENADOR
*
1140 REM ****************************
*
1150 FOR i=20 TO 35 STEP 5:SOUND 1,
i,1:NEXT i:w=0
1160 LOCATE 1,23:PRINT c$:PAPER #1,
0:LOCATE #1,1,8:PRINT #1,STRING$(8,
10)
1170 e=34:d=1:aa=o:o=1:GOSUB 3430:G
OSUB 1950:o=aa
1180 IF t=11 AND (v=1 OR v=10) THEN
 w=1:GOTO 1380
1190 IF bj1=1 AND p=0 THEN 1380
1200 IF bj1=1 AND bj2=1 THEN 1380
1210 IF s>21 AND (p=0 OR p>21 OR bj
2=1) THEN 1380
1220 IF t>s AND (p=0 OR p>21 OR bj2
=1) THEN 1380
1230 IF t>16 AND ((s>t AND s<20 AND
 (p=0 OR p>21)) OR (p>t AND p<20))
THEN no1=RND*7:no2=RND*10:IF no2<no
```




```
1 THEN 1380
1240 IF p>21 AND s=21 THEN 1380
1250 IF s>21 AND p=21 THEN 1380
1260 IF t>p AND (s>21 OR bj1=1) THE
N 1380
1270 IF ((t=p AND p>0) OR t=s) AND
t>11 THEN no1=RND*t:no2=RND*25:IF n
o2<no1 THEN 1380
1280 n=1
1290 o=o+1:e=n*2+1:GOSUB 3430:GOSUB
 1950:GOSUB 3450:t=t+a
1300 IF t>s AND bj1=0 THEN 1380
1310 IF t>p AND bj2=0 AND p>0 AND s
>21 THEN 1380
1320 IF t>16 AND s=21 THEN no1=RND*
10:no2=RND*35:IF no2>no1 THEN 1380
1330 IF t>16 AND ((s>t AND s<20) OR
 (p>t AND p<20)) AND (p=0 OR p>21)
THEN no1=RND*7:no2=RND*10:IF no2>no
1 THEN 1380
1340 IF t=s AND s>16 AND (p=0 OR p>
21 OR bj2=1) THEN 1380:'IF t=s AND
s>16 THEN 1350
1350 IF t=p AND p>16 THEN 1380:'IF
t=p AND p>16 AND (s>21 OR bj1=1) TH
EN 700
1360 IF t>21 THEN 1380
1370 n=n+1:GOTO 1290
1380 FOR i=150 TO 100 STEP -1:SOUND
 1,i,5:NEXT i
1390 GOSUB 1510
1400 IF bdna=1 THEN bdna=0:GOTO 145
0
1410 NEXT mano
1420 REM ---------------------------
----
1430 REM -AQUI TERMINA BUCLE PRINCI
PAL-
1440 REM ---------------------------
----
1450 GOSUB 3620
1460 CLS:INK 0,10:d$=CHR$(174)+"CON
TINUAMOS JUGANDO?":GOSUB 3370:GOSUB
 3500:IF x$="N" THEN CLS:SYMBOL AFT
ER 0:END
1470 LOCATE 1,10:PAPER 0:PRINT c$:C
LS:GOTO 350
1480 REM ****************************
*
1490 REM * RESULTADO DE LA PARTIDA
*
1500 REM ****************************
*
1510 PEN 2:CLS:PRINT CHR$(24);"
  -RESULTADO  DE  LA  PARTIDA-
 ";CHR$(24);
1520 IF w=1 THEN p$=" BLACK JACK" E
LSE p$=STR$(t)+" PUNTOS"
1530 x$="MI JUGADA HA SIDO"+p$:LOCA
TE 1,3:PRINT x$
1540 x$="SU JUGADA HA SIDO"
1550 IF p THEN 1580
1560 IF bj1=0 THEN x$=x$+STR$(s)+"
PUNTOS" ELSE x$=x$+" BLACK JACK"
1570 IF NOT p THEN LOCATE 1,5:PRINT
 x$:GOTO 1630
1580 LOCATE 1,5:PRINT x$;":"
1590 IF bj1=1 THEN vw$=" BLACK JACK
" ELSE vw$=STR$(s)+" PUNTOS"
1600 x$="PRIMER CAMINO "+CHR$(253)+
CHR$(252)+vw$:LOCATE 1,7:PRINT x$
1610 IF bj2=1 THEN wv$=" BLACK JACK
" ELSE wv$=STR$(p)+" PUNTOS"
1620 x$="SEGUNDO CAMINO "+CHR$(253)
+CHR$(252)+wv$:LOCATE 1,8:PRINT x$
1630 k=0:IF q=0 THEN 1650
1640 PRINT:IF w THEN PRINT "GANA EL
 SEGURO DE";q;"PTAS":k=k+2*q ELSE P
RINT "PIERDE EL SEGURO DE";q;"PTAS"
:k=k-q
1650 x=s:y=0
1660 PRINT
1670 IF p>0 AND y=0 THEN s$="1"+CHR
$(255)+" CAMINO:":PRINT s$;
1680 IF p>0 AND y=1 THEN q$="2"+CHR
$(254)+" CAMINO:":PRINT q$;
1690 IF x>21 THEN PRINT "PIERDE AL
PASAR DE 21":k=k-h:GOTO 1770
1700 IF w AND ((bj1=1 AND y=0) OR (
bj2=1 AND y=1)) THEN PRINT "EMPATAD
OS":GOTO 1770
1710 IF (bj1=1 AND y=0) OR (bj2=1 A
ND y=1) AND NOT w THEN PRINT "GANA
CON BLACK JACK":K=K+1.5*h:GOTO 1770
1720 IF w=1 AND ((bj1=0 AND y=0) OR
 (bj2=0 AND y=1)) THEN PRINT "PIERD
E CON MI BLACK JACK":k=k-h:GOTO 177
0
1730 IF t>21 THEN PRINT "GANA AL PA
SARME DE 21":k=k+h:GOTO 1770
1740 IF x>t THEN PRINT "GANA AL SUP
ERARME":k=k+h:GOTO 1770
1750 IF t>x THEN PRINT "GANO AL SUP
ERARLE":k=k-h:GOTO 1770
1760 PRINT "EMPATADOS"
1770 IF y=0 AND p THEN  x=p:y=1:GOT
O 1680
1780 PRINT:PRINT "EN ESTA MANO ";
1790 k=INT(k)
1800 IF k=0 THEN PRINT "QUEDAMOS EM
PATADOS":GOTO 1830
1810 IF k>0 THEN PRINT "GANA";k;"PT
AS":GOTO 1830
1820 PRINT "PIERDE";ABS(k);"PTAS"
1830 m=INT(m+k)
1840 PRINT:PRINT "TOTAL:";
1850 IF m=0 THEN PRINT "COMO AL PRI
NCIPIO" ELSE IF m>0 THEN PRINT m;"P
TAS GANADAS"ELSE PRINT ABS(m);"PTAS
 PERDIDAS"
1860 dg=dg+k
1870 PRINT:PRINT "TIENE";dg;"PTAS"
1880 IF dg<200 THEN PRINT:PRINT "YA
 NO PUEDE SEGUIR APOSTANDO":bdna=1
1890 WHILE INKEY$<>"":WEND:INK 1,0,
26:LOCATE 13,25:PEN 1:PRINT "PULSE
UNA TECLA":WHILE INKEY$="":WEND:PAP
ER 0
1900 INK 1,26
1910 RETURN
1920 REM ****************************
*
1930 REM *     DIBUJA LA CARTA
*
1940 REM ****************************
*
```

```
1950 GOSUB 3210
1960 ON a GOSUB 1970,2460,2490,2520
,2550,2580,2610,2640,2670,2700,2820
,2940,3060:RETURN
1970 ON c GOSUB 1980,2100,2220,2340
:RETURN
1980 RESTORE 2050
1990 FOR nv=3 TO 7
2000 FOR i=1 TO 5:FOR nn=1 TO 9:REA
D n(nn):NEXT nn:SYMBOL n(1),n(2),n(
3),n(4),n(5),n(6),n(7),n(8),n(9):NE
XT i
2010 LOCATE e+1,d+nv:PAPER 1:PEN b:
PRINT "!#%&'"
2020 NEXT
2030 GOSUB 3310
2040 RETURN
2050 DATA 33,0,0,0,0,0,0,1,3,35,0,0
,0,0,0,248,252,254,37,0,0,0,0,0,0,0
,0,38,0,0,0,0,0,31,63,127,39,0,0,0,
0,0,0,128,192
2060 DATA 33,7,15,15,31,31,63,63,63
,35,255,255,255,255,255,255,255,255
,37,0,129,195,231,255,255,255,255,3
8,255,255,255,255,255,255,255,255,3
9,224,240,240,248,248,252,252,252
2070 DATA 33,63,63,63,31,31,15,15,7
,35,255,255,255,255,255,255,255,255
,37,255,255,255,255,255,255,255,255
,38,255,255,255,255,255,255,255,255
,39,252,252,252,248,248,240,240,224
2080 DATA 33,3,1,0,0,0,0,0,0,35,255
,255,255,127,63,31,15,7,37,255,255,
255,255,255,255,255,255,38,255,255,
255,254,252,248,240,224,39,192,128,
0,0,0,0,0,0
2090 DATA 33,0,0,0,0,0,0,0,35,3,1
,0,0,0,0,0,0,37,255,255,255,126,60,
24,0,0,38,192,128,0,0,0,0,0,39,0,
0,0,0,0,0,0,0
2100 RESTORE 2170
2110 FOR nv=3 TO 7
2120 FOR i=1 TO 5:FOR nn=1 TO 9:REA
D n(nn):NEXT nn:SYMBOL n(1),n(2),n(
3),n(4),n(5),n(6),n(7),n(8),n(9):NE
XT i
2130 LOCATE e+1,d+nv:PAPER 1:PEN b:
PRINT "!#%&'"
2140 NEXT nv
2150 GOSUB 3310
2160 RETURN
2170 DATA 33,0,0,0,0,0,0,0,35,0,0
,0,0,0,0,0,0,37,24,24,60,60,126,126
,255,255,38,0,0,0,0,0,0,0,39,0,0,
0,0,0,0,0,0
2180 DATA 33,0,0,0,0,0,0,0,35,1,1
,3,3,7,7,15,15,37,255,255,255,255,2
55,255,255,255,38,128,128,192,192,2
24,224,240,240,39,0,0,0,0,0,0,0
2190 DATA 33,0,0,0,0,0,0,0,35,31,
31,63,127,127,63,31,31,37,255,255,2
55,255,255,255,255,255,38,248,248,2
52,254,254,252,248,248,39,0,0,0,0,0
,0,0,0
2200 DATA 33,0,0,0,0,0,0,0,35,15,
15,7,7,3,3,1,1,37,255,255,255,255,2
55,255,255,255,38,240,240,224,224,1
92,192,128,128,39,0,0,0,0,0,0,0
2210 DATA 33,0,0,0,0,0,0,0,35,0,0
,0,0,0,0,0,0,37,255,255,126,126,60,
60,24,24,38,0,0,0,0,0,0,0,39,0,0,
0,0,0,0,0,0
2220 RESTORE 2290
2230 FOR nv=3 TO 7
2240 FOR i=1 TO 5:FOR nn=1 TO 9:REA
D n(nn):NEXT nn:SYMBOL n(1),n(2),n(
3),n(4),n(5),n(6),n(7),n(8),n(9):NE
XT i
2250 LOCATE e+1,d+nv:PAPER 1:PEN b:
PRINT "!#%&'"
2260 NEXT nv
2270 GOSUB 3310
2280 RETURN
2290 DATA 33,0,0,0,0,0,0,0,35,0,1
,7,15,31,31,63,63,37,126,255,255,25
5,255,255,255,255,38,0,128,224,240,
248,248,252,252,39,0,0,0,0,0,0,0,0
2300 DATA 33,0,0,0,0,0,1,3,7,35,63,
63,31,15,7,251,253,254,37,255,255,2
55,255,255,255,255,255,38,252,252,2
48,240,224,223,191,127,39,0,0,0,0,0
,128,192,224
2310 DATA 33,15,31,31,63,63,63,63,6
3,35,255,255,255,255,255,255,255,25
5,37,126,255,255,255,255,255,255,25
5,38,255,255,255,255,255,255,255,25
5,39,240,248,248,252,252,252,252,25
2
2320 DATA 33,31,31,15,7,3,1,0,0,35,
255,255,255,255,254,252,248,0,37,25
5,219,153,24,24,60,60,60,38,255,255
,255,255,127,63,31,0,39,248,248,240
,224,192,128,0,0
2330 DATA 33,0,0,0,0,0,0,0,35,0,0
,0,0,3,15,112,0,37,60,126,126,126,2
55,129,0,0,38,0,0,0,0,192,240,14,0,
39,0,0,0,0,0,0,0,0
2340 RESTORE 2410
2350 FOR nv=3 TO 7
2360 FOR i=1 TO 5:FOR nn=1 TO 9:REA
D n(nn):NEXT nn:SYMBOL n(1),n(2),n(
3),n(4),n(5),n(6),n(7),n(8),n(9):NE
XT i
2370 LOCATE e+1,d+nv:PAPER 1:PEN b:
PRINT "!#%&'"
2380 NEXT nv
2390 GOSUB 3310
2400 RETURN
2410 DATA 33,0,0,0,0,0,0,0,35,0,0
,0,0,1,3,7,15,37,24,60,126,255,255,
255,255,255,38,0,0,0,0,128,192,224,
240,39,0,0,0,0,0,0,0,0
2420 DATA 33,0,0,0,0,0,1,1,3,35,31,
63,127,255,255,255,255,255,37,255,2
55,255,255,255,255,255,255,38,248,2
52,254,255,255,255,255,255,39,0,0,0
,0,0,128,192,192
2430 DATA 33,3,3,3,3,3,1,1,0,35,255
,255,255,255,255,255,255,255,37,255
,255,255,255,255,255,255,255,38,255
,255,255,255,255,255,255,255,39,192
,192,192,192,192,128,128,0
2440 DATA 33,0,0,0,0,0,0,0,35,255
,127,63,31,14,0,0,0,37,255,219,153,
24,60,60.60.60,38,255,254,252,248,1
```

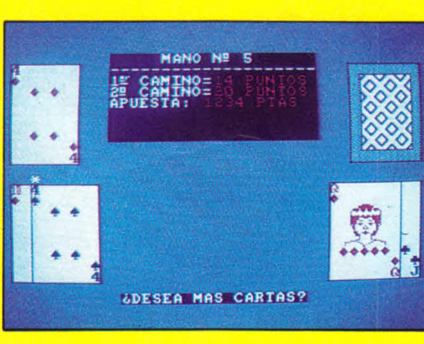



```
12,0,0,0,39,0,0,0,0,0,0,0,0
2450 DATA 33,0,0,0,0,0,0,0,35,0,0
,0,3,15,56,64,0,37,126,126,126,255,
192,0,0,0,38,0,0,0,192,240,12,2,0,3
9,0,0,0,0,0,0,0,0
2460 RESTORE 2730
2470 FOR i=1 TO 2:READ x,y
2480 LOCATE x+e,y+d:PAPER 1:PEN b:P
RINT MID$(b$,c,1):NEXT i:GOSUB 3310
:RETURN
2490 RESTORE 2740
2500 FOR i=1 TO 3:READ x,y
2510 LOCATE x+e,y+d:PAPER 1:PEN b:P
RINT MID$(b$,c,1):NEXT i :GOSUB 331
0:RETURN
2520 RESTORE 2750
2530 FOR i=1 TO 4:READ x,y
2540 LOCATE x+e,y+d:PAPER 1:PEN b:P
RINT MID$(b$,c,1):NEXT i:GOSUB 3310
:RETURN
2550 RESTORE 2760
2560 FOR i=1 TO 5:READ x,y
```




```
2570 LOCATE x+e,y+d:PAPER 1:PEN b:P
RINT MID$(b$,c,1):NEXT i:GOSUB 3310
:RETURN
2580 RESTORE 2770
2590 FOR i=1 TO 6:READ x,y
2600 LOCATE x+e,y+d:PAPER 1:PEN b:P
RINT MID$(b$,c,1):NEXT i:GOSUB 3310
:RETURN
2610 RESTORE 2780
2620 FOR i=1 TO 7:READ x,y
2630 LOCATE x+e,y+d:PAPER 1:PEN b:P
RINT MID$(b$,c,1):NEXT i:GOSUB 3310
:RETURN
2640 RESTORE 2790
2650 FOR i=1 TO 8:READ x,y
2660 LOCATE x+e,y+d:PAPER 1:PEN b::
PRINT MID$(b$,c,1):NEXT i:GOSUB 331
0:RETURN
2670 RESTORE 2800
2680 FOR i=1 TO 9:READ x,y
2690 LOCATE x+e,y+d:PAPER 1:PEN b:P
RINT MID$(b$,c,1):NEXT i:GOSUB 3310
:RETURN
2700 RESTORE 2810
2710 FOR i=1 TO 10:READ x,y
2720 LOCATE x+e,y+d:PAPER 1:PEN b:P
RINT MID$(b$,c,1):NEXT i:GOSUB 3310
:RETURN
2730 DATA 3,3,3,6
2740 DATA 3,3,3,5,3,7
2750 DATA 2,3,4,3,2,7,4,7
2760 DATA 2,3,4,3,2,7,4,7,3,5
2770 DATA 2,3,4,3,2,5,4,5,2,7,4,7
2780 DATA 2,3,4,3,3,4,2,5,4,5,2,7,4
,7
2790 DATA 2,3,4,3,3,4,2,5,4,5,3,6,2
,7,4,7
2800 DATA 2,3,3,3,4,3,2,5,3,5,4,5,2
,7,3,7,4,7
2810 DATA 2,3,4,3,2,4,4,4,2,5,4,5,2
,6,4,6,2,7,4,7
2820 RESTORE 2900
2830 FOR nv=3 TO 6
2840 FOR i=1 TO 5:FOR nn=1 TO 9:REA
D n(nn):NEXT nn:SYMBOL n(1),n(2),n(
3),n(4),n(5),n(6),n(7),n(8),n(9):NE
XT i
2850 LOCATE e+1,d+nv:PAPER 1:PEN b:
PRINT "!#%&'"
2860 NEXT nv
2870 LOCATE e+1,d+7:FOR j=0 TO 4:PA
PER 1:PEN b:PRINT MID$(b$,c,1);:NEX
T j
2880 GOSUB 3310
2890 RETURN
2900 DATA 33,0,0,0,0,0,0,0,35,255
,251,123,59,31,31,31,31,37,255,109,
109,109,255,224,224,227,38,255,190,
188,184,248,16,16,200,39,0,0,0,0,0,
0,0,0
2910 DATA 33,0,0,0,0,0,0,0,35,31,
31,31,31,31,31,63,63,37,225,224,224
,225,225,224,240,244,38,132,2,2,6,1
20,248,8,24,39,0,0,0,0,0,0,0
2920 DATA 33,0,0,0,0,0,1,14,16,35,6
3,16,16,15,16,228,2,113,37,227,1,0,
255,0,33,82,36,38,48,192,64,128,64,
62,1,112,39,0,0,0,0,0,128,120
2930 DATA 33,0,0,0,0,0,0,0,35,128
,68,131,0,0,0,0,0,37,0,0,1,222,32,0
,0,0,38,8,16,8,0,0,0,0,39,0,0,0,0
,0,0,0,0
2940 RESTORE 3020
2950 FOR nv=3 TO 6
2960 FOR i=1 TO 5:FOR nn=1 TO 9:REA
D n(nn):NEXT nn:SYMBOL n(1),n(2),n(
3),n(4),n(5),n(6),n(7),n(8),n(9):NE
```

```
XT i
2970 LOCATE e+1,d+nv:PAPER 1:PEN b:
PRINT "!#%&'"
2980 NEXT nv
2990 LOCATE e+1,d+7:FOR j=0 TO 4:PA
PER 1:PEN b:PRINT MID$(b$,c,1);:NEX
T
3000 GOSUB 3310
3010 RETURN
3020 DATA 33,0,0,0,0,0,0,0,0,35,0,3
,3,62,126,140,128,128,37,126,255,25
5,247,247,99,0,0,38,0,192,192,188,1
90,17,1,1,39,0,0,0,0,0,0,0,0
3030 DATA 33,0,0,0,0,0,0,0,0,35,214
,255,254,254,249,248,251,177,37,102
,255,60,0,195,0,195,129,38,107,255,
127,127,159,31,223,141,39,0,0,0,0,0
,0,0,0
3040 DATA 33,0,0,0,0,0,0,0,0,35,112
,48,48,16,28,28,15,15,37,24,36,60,0
,24,126,24,0,38,14,12,12,12,56,56,2
40,240,39,0,0,0,0,0,0,0,0
3050 DATA 33,0,0,0,0,15,8,8,0,35,1,
1,7,252,3,0,0,0,37,0,0,0,0,0,195,40
,48,38,128,128,208,63,192,0,0,0,39,
0,0,0,0,240,16,16,0
3060 RESTORE 3140
3070 FOR nv=3 TO 6
3080 FOR i=1 TO 5:FOR nn=1 TO 9:REA
D n(nn):NEXT nn:SYMBOL n(1),n(2),n(
3),n(4),n(5),n(6),n(7),n(8),n(9):NE
XT i
3090 LOCATE e+1,d+nv:PAPER 1:PEN b:
PRINT "!#%&'"
3100 NEXT nv
3110 LOCATE e+1,d+7:FOR j=0 TO 4:PA
PER 1:PEN b:PRINT MID$(b$,c,1);:NEX
T
3120 GOSUB 3310
3130 RETURN
3140 DATA 33,0,0,0,0,0,0,0,0,35,68,
78,123,63,31,15,62,61,37,33,115,222
,255,115,255,24,193,38,9,157,247,25
4,156,248,62,222,39,0,0,0,0,0,0,0,0
3150 DATA 33,0,0,0,0,0,0,0,0,35,56,
27,25,24,24,12,14,15,37,34,128,128,
0,0,34,28,255,38,14,236,204,12,12,2
4,56,248,39,0,0,0,0,0,0,0,0
3160 DATA 33,0,15,57,73,136,136,136
,128,35,7,255,193,192,192,64,64,0,3
7,255,227,255,255,127,62,0,0,38,240
,255,195,131,3,2,2,0,39,0,240,156,1
46,18,18,18,2
3170 DATA 33,4,2,1,0,0,0,0,0,35,0,2
4,25,129,65,63,0,0,37,0,0,0,0,0,255
,24,0,38,0,24,152,129,130,252,0,0,3
9,16,32,64,128,0,0,0,0
3180 REM ****************************
*
3190 REM *FONDO Y MARCO DE LA CARTA
*
3200 REM ****************************
*
3210 FOR i=1 TO 9:LOCATE e,d+i:PAPE
R 1:PRINT "          ":NEXT i
3220 f=e*16-16
3230 g=399-d*16+2
3240 PLOT f,g,2
3250 DRAWR 110,0:DRAWR 0,-146:DRAWR
 -110,0:DRAWR 0,146
3260 PEN 3
3270 RETURN
3280 REM ****************************
*
3290 REM * IDENTIFICADOR DEL NAIPE
*
3300 REM ****************************
*
3310 PRINT CHR$(22);CHR$(1);:PAPER
1:PEN b:LOCATE e,d+1:PRINT MID$(a$,
a,1):LOCATE e,d+2:PRINT MID$(b$,c,1
):LOCATE e+6,d+9:PRINT MID$(a$,a,1)
:LOCATE e+6,d+8:PRINT MID$(b$,c,1):
PRINT CHR$(22);CHR$(0);:GOSUB 3240
3320 PAPER 0
3330 RETURN
3340 REM ****************************
*
3350 REM *   IMPRIME EL MENSAJE
*
3360 REM ****************************
*
3370 LOCATE 1,23:PAPER 0:PRINT c$:P
APER 2:PEN 1:LOCATE 21-LEN(d$)/2,23
:FOR i=1 TO LEN(d$):PRINT MID$(d$,i
,1);:NEXT i
3380 PAPER 0
3390 RETURN
3400 REM ****************************
*
3410 REM *  EXTRAE CARTA DEL MAZO
*
3420 REM ****************************
*
3430 c=INT((ASC(MID$(e$,o,1))-1)/13
)+1:a=ASC(MID$(e$,o,1))-13*(c-1):IF
 c>2 THEN b=2 ELSE b=3
3440 RETURN
3450 IF a>10 THEN a=10
3460 RETURN
3470 REM ****************************
*
3480 REM *  ESPERA PULSAR DE TECLA
*
3490 REM ****************************
*
3500 et$=INKEY$:WHILE et$<>"":FOR r
```




```
et=1 TO 50:NEXT:et$=INKEY$:WEND
3510 x$="":WHILE x$<>"S" AND x$<>"N
":x$=UPPER$(INKEY$):WEND
3520 RETURN
3530 REM ****************************
*
3540 REM *  SUBRUTINAS AUXILIARES
*
3550 REM ****************************
*
3560 PLOT 3*16-16,400-12*16:DRAWR 1
6,0:RETURN
3570 PLOT 38*16-16,400-12*16:DRAWR
16,0:RETURN
3580 PLOT 35*16-16,400-9*16,2:DRAWR
 5*16,0:DRAWR 0,7*16:DRAWR -5*16,0:
DRAWR 0,-7*16:RETURN
3590 REM ****************************
3600 REM *  MAXIMAS PUNTUACIONES  *
3610 REM ****************************
3620 MODE 1:INK 3,24:INK 1,6:PEN 3:
PAPER 0:IF dg<=rec(6) THEN 3830
3630 LOCATE 6,5:PRINT "ES UNO DE LO
S MAS RICOS DE HOY"
3640 PLOT 6*16-16-8,400-5*16-8,2:DR
AWR 30*16+16,0:DRAWR 0,32:DRAWR -30
*16-16,0:DRAWR 0,-32
3650 can$="HA TERMINADO CON"+STR$(D
G)+" PTAS"
3660 can=LEN(can$):IF can MOD 2=0 T
HEN can$=" "+can$
3670 PEN 2:LOCATE 20-can/2,10:PRINT
 can$
3680 MOVE 12*16-8,400-16*16+8:TAG:P
RINT "ESCRIBA SU NOMBRE";:TAGOFF
3690 WINDOW #1,12,29,18,20:PAPER #1
,2:PEN #1,1:CLS#1:LOCATE #1,6,2:PRI
NT #1,"........";STRING$(13,8);
3700 PLOT 12*16-16-2,400-20*16-2,3:
DRAWR 17*16+16+2,0:DRAWR 0,50::DRAW
R -17*16-16-2,0:DRAWR 0,-50
3710 WHILE INKEY$<>"":WEND
3720 te$="":nom$="":WHILE te$<>CHR$
(13)
3730 IF te$>CHR$(31) AND te$<"z" AN
D LEN(nom$)<8 THEN nom$=nom$+te$:PR
INT #1,te$;
3740 IF te$=CHR$(127) AND LEN(nom$)
 THEN nom$=LEFT$(nom$,LEN(nom$)-1):
PRINT #1,CHR$(8);CHR$(16);".";CHR$(
8);
3750 te$=INKEY$
3760 WEND
3770 PAPER 0
3780 rec$(6)=nom$
3790 rec(6)=dg
3800 FOR i=6 TO 2 STEP -1
3810 IF rec(i)>rec(i-1) THEN ks$=re
c$(i):rec$(i)=rec$(i-1):rec$(i-1)=k
s$:ks=rec(i):rec(i)=rec(i-1):rec(i-
1)=ks
3820 NEXT i
3830 CLS
3840 PLOT 0,0,2:DRAWR 638,0:DRAWR 0
,398:DRAWR -638,0:DRAWR 0,-398
3850 PAPER 2:PEN 3:LOCATE 8,1:PRINT
 "LOS MAS RICOS DE HOY SON :"
3860 FOR i=1 TO 6:PEN 1:PAPER 3:LOC
ATE 12,i*2+1:PRINT USING "\      \"
;rec$(i);:PAPER 0:PEN 2:WHILE POS(#
0)<24:PRINT ".";:WEND:PEN 2:PAPER 3
:PRINT USING "######";rec(i)
3870 NEXT
3880 FOR i=0 TO 640 STEP 18:PLOT i,
0,2:DRAW 0,400-i/1.6:NEXT
3890 FOR i=0 TO 640 STEP 18:PLOT i,
0,2:DRAW 640,i/1.6:NEXT
3900 LOCATE 12,18:PEN 1:PAPER 2:PRI
NT " PULSE UNA TECLA. ":LOCATE 12,1
7:PRINT STRING$(18,32);:LOCATE 12,1
9:PRINT STRING$(18,32)
3910 WHILE INKEY$<>"":WEND:WHILE IN
KEY$="":WEND
3920 PAPER 0:PEN 1
3930 CLS:INK 0,10:INK 1,26:INK 2,0:
INK 3,6
3940 RETURN
3950 REM ****************************
**
3960 REM *      ESCRIBE VENTANA
 *
3970 REM ****************************
**
3980 LOCATE #1,6,1:PEN #1,1:PRINT #
1,"MANO N";CHR$(254);:PRINT #1,USIN
G "##";mano:LOCATE #1,1,2:PRINT #1,
STRING$(20,"-")
3990 PEN #1,1:LOCATE #1,1,3:PRINT #
1,"1";CHR$(255);" CAMINO=";:PEN #1,
3:IF bj1=1 THEN PRINT#1,"BLACK JACK
":FOR ret=1 TO 300:NEXT ELSE PRINT
#1,USING "##";s;:PRINT #1," PUNTOS"
4000 IF p THEN PEN #1,1:LOCATE #1,1
,4:PRINT #1,"2";CHR$(254);" CAMINO=
";:PEN #1,3:IF bj2=1 THEN PRINT #1,
"BLACK JACK":FOR ret=1 TO 300:NEXT
ELSE PRINT #1,USING "##";p;:PRINT #
1," PUNTOS"
4010 PEN #1,1:LOCATE #1,1,5:PRINT #
1,"APUESTA:";:PEN #1,3:PRINT #1,USI
NG "#####";h;:PRINT #1," PTAS"
4020 IF q THEN PEN #1,1:LOCATE #1,1
,6:PRINT #1,"SEGURO:";:PEN #1,3:PRI
NT #1,USING "####";q;:PRINT #1," PT
AS"
4030 IF r THEN LOCATE #1,1,7:PEN#1,
3:PRINT #1,"APUESTA DOBLADA"
4040 RETURN
```

Para que tus dedos no realicen el trabajo duro, M.H. AMSTRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicítanoslo.

RUN "TASWORD"
© 1984 TASMAN SOFTWARE LTD.
TASMAN SOFTWARE

Tasman SOFTWARE

por fin en España, software a precios británicos

TASWORD

¿Se imagina su ordenador convertido en una máquina de escribir? TASWORD es la mejor relación calidad-precio en tratamiento de texto profesional.

Totalmente en castellano, permitiendo realizar MAIL MERGE, trabajar en bloques sin ninguna interrupción incrementando su velocidad, etc... (en versión 6128 aprovecha las 128 K creando un disco virtual de 64 K).

- Acentos, ñ, ü, ?, etc...
- Compatible Productos TASMAN.
- Adaptación impresoras.
- Configuración propia por usuario.
- Ensamblaje de textos.

9.900 pts. AMSTRAD COMMODORE EINSTEIN MSX

6.900 pts. AMSTRAD COMMODORE MSX SPECTRUM

7.900 pts. SPECTRUM

TAS-SPELL

Primer auxiliar que corregirá la ortografía de sus escritos y pondrá los acentos olvidados no dando margen a ningún error.

Contiene un potente diccionario con más de 20.000 vocablos pudiendo Vd. ampliarlos. Complemento ideal para su TASWORD con disco.

7.600 pts. AMSTRAD



Próximamente en versión PCW 8256 8512

TAS-PRINT

Con TAS-PRINT la escritura elevada a arte. Utiliza las grandes posibilidades gráficas de su ordenador. Las posibilidades tipográficas las explota al máximo al dar una doble pasada optimizando la calidad.

Los tipos de escritura son: COMPACTA MEDIAN DATA-RUN LECTURA LIGHT PALACE SCRIPT

7.600 pts. AMSTRAD EINSTEIN

5.900 pts. AMSTRAD SPECTRUM

6.900 pts. QL SPECTRUM

TASCOPY

Sin necesidad de un PLOTTER podrá obtener sus gráficos de pantalla a través de la impresora.

Un increíble ZOOM le permite realizar sus gráficos en 4 hojas formando un póster de gran tamaño.

7.600 pts. AMSTRAD

5.900 pts. AMSTRAD SPECTRUM

6.900 pts. QL SPECTRUM

GRAFMAN

Programa de E.G. Computer Graphics especialmente diseñado para trabajar conjuntamente con TASCOPY representando las funciones matemáticas en desarrollos de diagramas por coordenadas, permitiendo su efecto "ZOOM" ampliar sectores de dichos diagramas.



5.600 pts. 6.200 pts. SOLO AMSTRAD

TOTALMENTE EN ESPAÑOL

- IVA NO INCLUIDO

DE VENTA EN LOS MEJORES COMERCIOS DE INFORMATICA

Si Vd. tiene alguna dificultad para obtener los programas, puede dirigirse a:

Ofites Informática

Avda. Isabel II, 16 - 8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

CONDICIONES ESPECIALES PARA DISTRIBUIDORES

EDITOR Y DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA Y PORTUGAL

EL MUNDO DE BLOQUES

Uno de los temas más espinosos de la IA, todavía no resuelto, es lo que los investigadores en este terreno llaman «proceso del lenguaje natural» o «comprensión del lenguaje natural», es decir, conseguir que los ordenadores comprendan el lenguaje humano.

Creo que todos podemos imaginar la complejidad que este propósito entraña, ya que nuestros idiomas están llenos de ambigüedades, frases hechas y coloquiales, etc, que nosotros mismos comprendemos sólo gracias al enorme banco de memoria que tenemos en nuestro cerebro, y a algo indefinible, por ahora al menos, que llamamos inteligencia.

Los computadores no tienen de eso, pero, como ya hemos aprendido en el curso de IA, pueden simularlo de manera bastante convincente (recuerden el programa Eliza).

En el caso del lenguaje, éste sólo puede ser comprendido dentro de un entorno predecible, esto es, que posea una serie de posibilidades o «movimientos» fijos.

Hubo un hombre, Terry Winnograd, que marcó un hito hasta ahora insuperado con un programa, escrito en «Planner», en la comprensión del lenguaje natural; podía mantener conversaciones con su «obra» de una increíble complejidad. Nosotros, con algunas lógicas limitaciones, hemos escrito en Basic una versión del Mundo de Bloques, y, lo que es mucho más decisivo, hemos explicado con todo detalle las técnicas que se usan para que una máquina comprenda nuestra lengua y actúe en consecuencia.

Sin más preámbulos, AMSTRAD Semanal presenta... El Mundo de Bloques.

Por José Antonio Morueco González





La inteligencia artificial es un área de la informática muy importante, que nos permite atacar tareas que no podían ser realizadas hasta hace pocos años. La inteligencia artificial tiene unas técnicas propias especiales y muy variadas según sea el objetivo a conseguir. Estos objetivos se dividen en tres grandes grupos:

— Los sistemas expertos.

— Los procesadores de lenguaje natural.

— La robótica.

Tanto la parte de los sistemas expertos como la de la robótica no tienen mucho que ver con el programa aquí presentado, aunque sí es bueno saber un poco sobre su objetivo primordial: ayudar al hombre a resolver situaciones y a tomar decisiones basándose en el conocimiento «inteligente» que tiene el ordenador en el área del saber en la que estamos. Este conocimiento inteligente, su presentación y su gestión, es el que distingue la inteligencia artificial del resto de las técnicas informáticas.

Lenguaje natural

Por otra parte, que nos atañe más en este artículo, está el procesamiento del lenguaje natural. Para una persona que no esté familiarizada con el mundo de la programación, este problema puede parecerle no muy compilado, porque a él le cuesta muy poco comprender lo que se dice en un escrito, o lo que se dice por la radio. Pero el problema de hacerle entender al ordenador nuestro lenguaje natural, aun teniendo ciertas estructuras, es muy poco sumiso a unas reglas de construcción severas. Esta flexibilidad del lenguaje humano es la que complica las cosas al ordenador.

Los lenguajes con los que el hombre se comunica con el ordenador son de unas estructuras muy severas, y en los que hay pocas palabras distintas, aunque desde fuera pueda parecer todo lo contrario. Cuando una persona hace un programa en BASIC, por ejemplo, tiene que andarse con mucho cuidado de utilizar exactamente las palabras reservadas que hagan falta; en caso contrario tendrá muchos errores. Cuando hablamos con un amigo no es necesaria tanta precisión, y de hecho no pensamos qué palabra usar en esta frase, o qué estructura sintáctica en aquella otra: simplemente decimos lo que se nos ocurra.

Todo esto nos conduce a reflexionar sobre cómo hacer que podamos conseguir que el ordenador comprenda castellano directamente. Para ello es necesario hacer un programa que sirva de interfase entre nuestras palabras y la acción que se quiere realizar. Esta interfase es la que traduce, a código más comprensible para el ordenador, dichas palabras.

Hasta ahora, no se han hecho procesadores de lenguaje natural que puedan considerarse completos, sino que se tiene un conjunto de palabras limitado o palabras claves, que son las que entiende el programa y mediante las cuales intenta comprender las frases dadas, como se verá luego más prácticamente con el programa que se presenta.

Claves del procesamiento del lenguaje natural

El procesamiento de lenguaje natural tiene varios puntos fundamentales:

— El diccionario, o conjunto de palabras que son conocidas y pueden traducirse.



— Las reglas sintácticas y estructuras del lenguaje natural que son analizadas por el procesador.

— Las herramientas que pueden usarse, etc.

Con respecto a las herramientas, es muy importante hacer notar que el BASIC no es precisamente muy adecuado para la inteligencia artificial, lo que produce en algunos casos ciertas limitaciones; se están desarrollando lenguajes, como LISP o PROLOG, más adecuados para estas cuestiones, y permiten unas representaciones de las estructuras más satisfactorias.

De todas formas esto no es un obstáculo insalvable; de hecho, pienso que es interesante ver, en un lenguaje de programación de gran extensión, algunas de estas técnicas, aunque la forma de representación no sea la más óptima.

En el caso que nos ocupa, tenemos un conjunto limitado de palabras que realmente traduce el programa, pero a la hora de usarse parece que hay más, debido a que puden usarse muchas más palabras, aunque sean ignoradas en la comprensión de la frase. Pasemos pues al programa.

Un procesador de lenguaje natural

¿Qué se puede hacer con el programa? Antes de ver cómo funciona, veamos un poco cómo se usa. Cuando des al «run», para ejecutar el programa, te aparecerán varios cuadros y triángulos de distintos colores y tamaños, y debajo de éstos, unas líneas de texto.

Ahí es donde debes ir dándole las órdenes al programa. Esta órdenes consiten en cuál de las figuras quieres que se mueva, y dónde. Por ejemplo puedes escribir:

pon el triángulo pequeño rojo encima del cuadro grande azul

Como puedes observar, tienes también unos números debajo de las figuras, del 1 al 12, que también pueden usarse; por ejemplo, después del movimiento anterior, la posición donde estaba el triángulo pequeño rojo, la número 10, queda vacía. Por ello, puedes ahora llevar allí otra figura; por ejemplo puedes dar la orden:

lleva el cuadro pequeño verde a la posición 10

De esta forma puedes ir realizando los movimientos que desees de una forma natural. Así, por ejemplo, el primer movimiento lo puedes expresar de muchas maneras diferentes en castellano, igualmente comprensible para el programa; algunas de estas formas podrían ser:

pon sobre el cuadro azul grande el triángulo rojo pequeño apila el triángulo rojo pequeño en el cuadro azul grande
sitúa debajo del triángulo pequeño rojo el cuadro grande azul

Como ves, hay bastantes variaciones ideomáticas para expresar una cosa tan sencilla como ésta. Este es uno de los problemas de traducir por ordenador el lenguaje natural, como decíamos en la introducción. Debido a ello hay ciertas limitaciones, pero bastante razonables. Por ejemplo, no intentes comprender frases que pueden ser ambiguas en su significado, ni frases tratadas de forma poética. —como podrían ser «pon el gran cuadro rojo sobre el triángulo cielo»— cosa que complicaría más todavía la lógica del programa, ya de por sí complicada sólo con un lenguaje normal.

Las leyes semánticas del programa «bloques»

Aparte de las cuestiones sintácticas, también he introducido algunas reglas semántias sencillas como son:

— No puedes poner figuras del mismo color apiladas juntas.

— No poner nada encima de un triángulo, porque, visto con naturalidad, se caería.

— Algo grande no pued ir encima de algo más pequeño, cosa también razonable, etc.

Estas reglas tienen por misión que se vea cómo se entiende la frase y, o bien se lleva a cabo el movimiento indicado, si es posible, o bien se indica el porqué no se puede hacer.

Para que lo veas, haz todas las combinaciones posibles de movimientos, e irás viendo que hay muchos posibles.

Por último notar que para acabar basta con decirle «adiós».

Visto ya cómo usarlo entremos un poco en su construcción. La idea general consiste en adivinar qué palabras clave están en la frase del usuario y cuál es la estructura de la frase.

Representación Del conocimiento

Cuando empiezas a pensar en cómo construir un programa así, lo primero que hay que limitar es el número de palabras y estructuras que van a ser comprendidas. En nuestro caso tenemos 5 estructuras generales, las cuales son bastante razonables.

Estas estructuras son:

1. Poner el objeto A encima del objeto B.

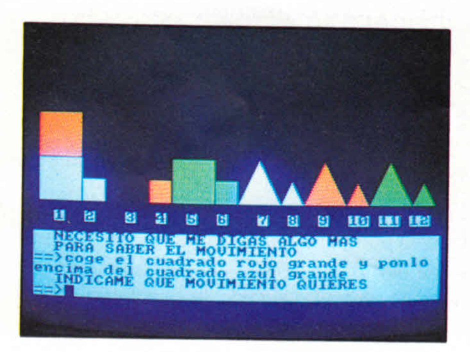
2. Poner el objeto B debajo del objeto A.

3. Poner encima del objeto B el objeto A.

4. Poner debajo del objeto A el objeto B.

5. Poner el objeto A en la posición n (con n entre 1 y 12).

donde el objeto puede ser un triángulo o un cuadrado, de un cierto tamaño (grande o pequeño), y un cierto color (verde, azul o rojo).



Cada objeto, junto con sus atributos, se considerará como un todo a la hora de analizar la frase. También hay que notar que llamo objeto A al que está encima, y objeto B al que va a estar debajo; esto se usará al hacer el programa.

En el programa hay varias partes, que iremos viendo:

— El programa principal, es la mayoría del programa, y donde se va analizando la estructura de la frase y, una vez hecho esto, se llevan a cabo las acciones semánticas (movimiento a hacer o mensaje de por qué no hacer el moviento).

— Las subrutinas 2500 y 3000, cuya misión es dibujar, o borrar, cuadrados y triángulos, respectivamente.

— La subrutina 2000 es clave y tiene por misión decir en qué posición está una palabra (pal$) en una frase (fr$), si está.

Vayamos al programa principal línea a línea. Para comprenderlo, veamos primero las variables más importantes y su función:

— px(i),py(i) —con i entre 1 y 12— tienen la posición horizontal y vertical de los objetos. Estos objetos están numerados del 1 al 12 y son:

1=cuadrado azul grande.
2=cuadrado azul pequeño.
3=cuadrado rojo grande.
4=cuadrado rojo pequeño.
5=cuadrado verde grande.
6=cuadrado verde pequeño.
7=triángulo azul grande.
8=triángulo azul pequeño.
9=triángulo rojo grande.
10=triángulo rojo pequeño.
11=triángulo verde grande.
12=triángulo verde pequeño.

En las primeras líneas se inicializan estas posiciones para la configuración inicial. También, asociada a éstas, está p(i), que mantiene siempre las posiciones iniciales de px(i), y sirve para los números que aparecen en pantalla del 1 al 12, y para cuando se indique que se ha de poner algo en dichas posiciones.

— db(i), indica qué figura está debajo de la i-esima; si no hay ninguna valdrá 0. Por ejemplo, si el triángulo rojo pequeño está encima del cuadrado azul grande entonces db(10)=1.

— oc(j)— con j entre 1 y 6 —indica si el cuadrado número j está cubierto— 0c(j)=—1— o si está descubierto —oc(j)=0.

— La variable p va a usarse para posiciones en frases donde está cierta palabra, según se calcula en la subrutina 2000. Además hay variables pr, pa y pb que indican posiciones en la frase de la preposición, el objeto A y el objeto B. Está también la varible n, que indica la posición de la frase donde se ha de empezar a buscar la palabra.

— na y nb contienen al final los números asociados a los objetos A y B respectivamente —números del 1 al 12.

— Otra variable importante es hn que indica si la estructura de la frase es la quinta, es decir, en la que se indica un movimiento a una posición numérica, en cuyo caso hn vale —1. En caso contrario valdrá 0.

— También está la variable cs, que vale 1 para las estructuras 1 y 4, en las cuales el objeto A está delante, en la frase, del objeto B; y vale 2 en las estructuras 2 y 3, que ocurre lo contrario.

— La variable ar indica si la estructura usa preposiciones como en, encima de, sobre, en cuyo caso ar vale —1, o si se usan preposiciones como bajo, debajo, en cuyo caso vale 0. Es decir, es otra variable para discernir la estructura.

— La variable aux indica si el objeto A es un triángulo —aux=6— o es un cuadrado —aux=0.

— La variable t indica si el objeto es grande o pequeño, según valga 32 ó 16; t es la apotema del objeto. Este valor depende de si el número del objeto es par —pequeño— o impar —grande.

— Para indicar el color están las variables ca y cb.

Hay alguna variable más, pero éstas son las más importantes.

Las demás las iremos viendo cuando aparezcan.

Veamos ya la realización del programa.

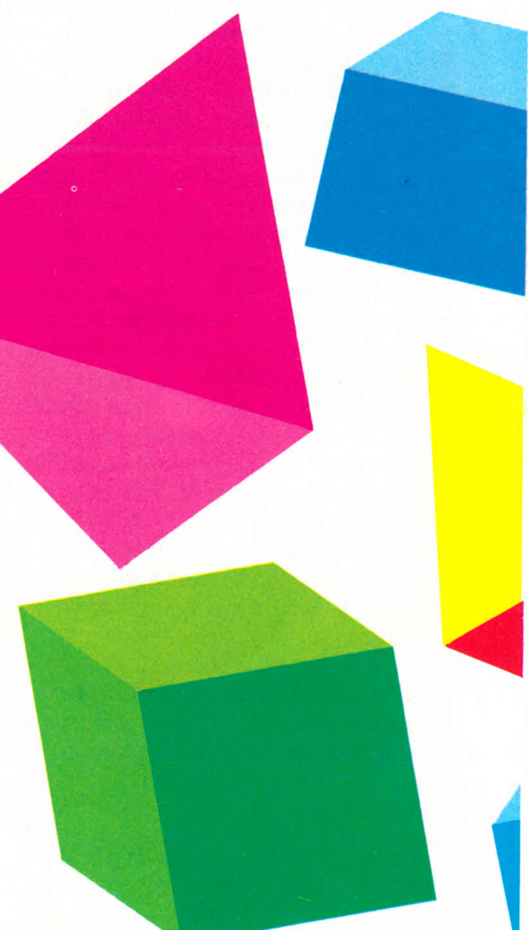
* Las primeras líneas, hasta la 280, es la inicialización de las variables, dibujar en pantalla el estado inicial, con el control de colores conveniente, la construcción de la ventana de texto.

Esta parte es sencilla y su única complicación es el cálculo de las posiciones en pantalla en donde se han de dibujar los objetos.

Después se escribe el texto inicial y, en la línea 300, se pide ya el movimiento deseado, que se mete en la variable fr$.

Se mira si la palabra es «adiós» para terminar.

A partir de aquí, se empieza ya el análisis de la frase con la intención de saber al final del análisis el valor de la na y nb, que representan el número del objeto de arriba y abajo.



Entre las líneas 330 y 370 se mira si hay en la frase algún número del 1 al 12. Si lo hay estamos en la quinta estructura y hacemos hn=—1. Si no la hay hecemos hn=0.

Si hemos encontrado el número, que será el ji, miramos —líneas 410 a la 450— si dicha posición ji está ocupada; esto ocurrirá cuando exista algún j con p(ji)=px(j), es decir, cuando algún objeto tenga su posición horizontal en la misma de ji. Si está ocupada, entonces se manda el mensaje correspondiente y se vuelve a empezar con otra orden. En caso contrario, se va a la línea 620, para ver qué objeto es el que ha de llevar a la posición ji.

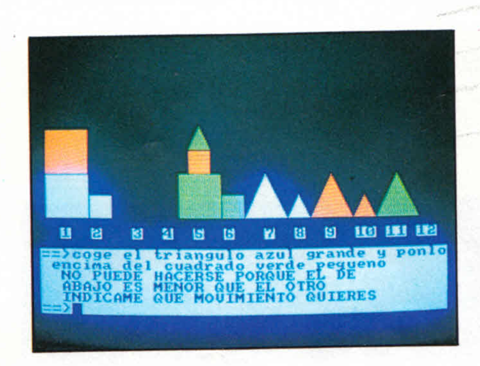
Si hn=0 entonces sigue por las líneas 460 hasta la 550, en donde se va viendo cuál es la preposición que se usa en la frase. Según cual sea, ar valdrá —1 ó 0, según vimos al ver las variables. Si no se encuentra ninguna de estas preposiciones se manda el mensaje correspondiente y se empieza de nuevo. En la 610 ponemos en pr la posición donde estaba dicha preposición.

En las 620-630 se comprueba si la palabra cuadrado está en la frase. Si no está y no estamos en la quinta estructura, entonces es imposible porque el objeto de abajo tiene que ser un cuadrado y se manda el mensaje conveniente de la línea 650.

En caso de la quinta estructura, si hn, es posible que no esté ningún cuadrado en la frase, y será un triángulo. Si no estamos en la quinta estructura, entonces hacemos pb=p, lo cual es provisional, y buscamos en la frase si hay, después de el cuadro encontrado, otro cuadro —por eso tomo n=pb+8, posición posterior a la de cuadrado. Si lo hay, no hace falta buscar si está el triángulo y nos saltamos las líneas 710-740; y variamos las posiciones pa y pb, suponiendo el caso 1. Las posiciones se cambiarán o se dejarán según la estructura que tengamos.

Si no hay dos cuadrados se mira si hay triángulo, como antes con los cuadrados; si no lo hay se manda el mensaje como pa, y aux=6, como se vio al ver las variables.

Así ya tenemos en pa y pb las posiciones de los dos objetos. En caso de la quinta estructura no es necesario y simplemente no influirá en su ejecución dichos parámetros.



Una parte fundamental está entre las líneas 750-760, en donde se analiza la estructura en la que nos encontramos. En las líneas 750 y 753, por medio de dos instrucciones condicionadas, se consigue saber cuál es el valor de cs:

— Si pr < pa y pr < pb y ar=1 entonces estaremos en la estructura 2 con lo que cs=2.

— Si pr < pa y pr < pb y ar=0 entonces estaremos en la estructura 4 con lo que cs=1.

— Si no se da la condición de las posiciones y ar=—1, estaremos en la estructura 3 y cs=2.

Tras esto sabremos ya el orden del objeto A y el objeto B:

cs=1: el objeto A delante del objeto B.

cs=2: el objeto B delante del objeto A.

Recordar que el objeto A es el de arriba y el objeto B es el de abajo, por lo cual es muy importante saber su orden. Todos estos cálculos se consiguen viendo todos los casos posibles que se pueden presentar en las extructuras, y, por ello, es lo más complicado de entender.

Después, en la línea 756, se comprueba si el objeto B, el de abajo, es un triángulo, por medio de un if un poco complicado, pero sólo ve las posibilidades de que así sea; en dicho caso nos vamos a la línea 650, donde se da el nombre conveniente.

Por último, en la línea 758, si cs=2, es decir, si estamos en una estructura con el objeto B delante del objeto A, se intercambian pa y pb, para que tengan ya la posición de A y B de forma correcta, respectivamente.

Entre las líneas 760 y 950 se calcula el color que tiene cada objeto A y B actuales. Para ello, se usa un data con las palabras azul, rojo y verde, y sus números asociados, 1, 2 y 3. Si estamos con hn=—1 entonces es más sencillo porque sólo se busca un color.

Veamos qué pasa en los demás casos. Para ello se toman dos subpartes de la frase total:

— fr$(0) es la correspondiente al objeto A.

— fr$(1) es la correspondiente al objeto B.

Estas se calculan según estemos en unas estructuras o en otras, dependiendo del valor de cs y vía unos cálculos sencillos de las posiciones pb y pa. Después se van leyendo de los datas estos colores para cada subfrase adecuada, obteniendo en n(0) el color de A, y en n(1) el de B, valores que se traspasan a ca y cb respectivamente. Si no se encuentran estos colores se saca el mensaje adecuado, saltando en la línea 910 a la 720.

Si los colores coinciden, también se lanza el mensaje adecuado. En el caso de hn=—1 sólo tenemos, como es de esperar, un sólo color en ca.

Entre las líneas 960-1080, se calcula, de forma parecida, los tamaños de los objetos A y B, los cuales están al final en n(0) y n(1) respectivamente. El procedimiento es similar al anterior, incluido para hn=—1.

Ahora, calculo ya los valores de na y nb, que son los números asociados a los objetos A y B que se han calculado. Si estamos en la estructura 5, hn=—1, nb=0 y no hay problemas. En la línea 1110 se calcula, en caso de la estructura 5, si el objeto cabe en la posición en donde se le intenta colocar.

En estas líneas se comprueba también si el cuadrado de abajo, objeto B, estaba ya ocupado, oc(nb)=—1, o si el de arriba está ya cubierto. En ambos casos el movimiento no se hace y se manda el mensaje conveniente. También se ha comprobado si el objeto de arriba no es mayor que el de abajo.

Lo que queda de programa principal es borrar el objeto A de donde esté y colocarlo sobre el objeto B. Para borrarlo se va a la subrutina 2500 ó 3000, según sea cuadrado o triángulo, y con br=—1 se pinta con el color de fondo dicho objeto. Para dibujarlo en su sitio actual se calcula, según hn y según el objeto sea grande o pequeño, los valores de las nuevas posiciones; la variable xx se usa para saber, antes de borrar el objeto, si va a caber, en la nueva posición, dentro de la pantalla.

Por último poner debajo del objeto A el objeto b —db(na)=nb —y desocupar el anterior, lo cual se hace antes, como debe ser. Ya sólo decir que las subrutinas de dibujo son más sencillas. Sólo indicar que la variable p indica el color.

Conclusión:

Espero que hayas entendido las ideas de la construcción del programa y que te sirva para que veas lo complicado que resulta llegar a comprender, digitalmente, el lenguaje humano.

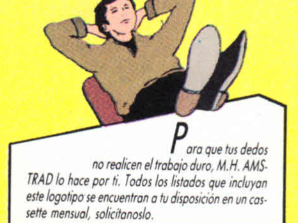
También es importante notar que el BASIC no es muy adecuado, com ya quedó dicho y como se demuestra al seguir el programa paso a paso.



```
10 DIM PX(12),PY(12),p(12),DB(12)
20 MODE 1
30 INK 0,11:INK 1,0:INK 2,9:INK 3,6
40 BORDER 0
50 PAPER 1: CLS
60 PAPER 0:WINDOW #1,1,40,20,25
70 FOR I=1 TO 6:PRINT #1:NEXT
80 PX(1)=43:P(1)=43
90 FOR I=2 TO 12
100 PX(I)=PX(I-1)+52
110 P(I)=PX(I)
120 NEXT I
130 FOR I=1 TO 12
140 IF I MOD 2=0 THEN PY(I)=161 ELSE PY(I)=177
150 NEXT I
160 FOR I=1 TO 6
170 GOSUB 2500
180 NEXT I
190 FOR I=7 TO 12
200 GOSUB 3000
210 NEXT I
220 FOR J=1 TO 12
230 K=K+3
240 IF K=9 OR K=22 THEN K=K+1
250 LOCATE K,18:PRINT MID$(STR$(J),2,2)
260 NEXT J
```

```
==>coge el cuadrado rojo pequeno y ponlo encima del cuadrado rojo grande
ESE MOVIMIENTO NO LO PUEDES HACER
PORQUE SON DEL MISMO COLOR
INDICAME QUE MOVIMIENTO QUIERES
==>
```

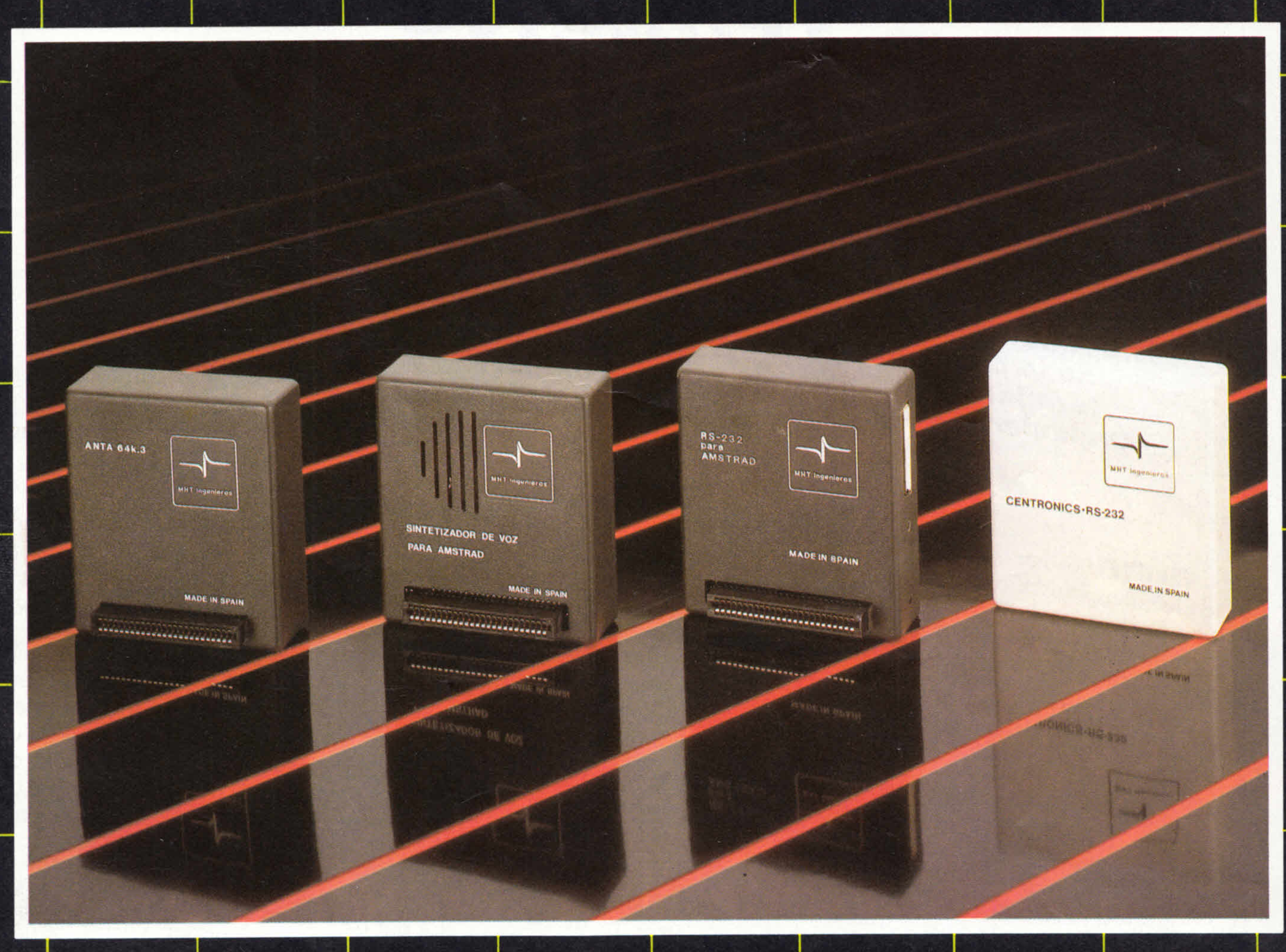
```
270 PRINT #1,"  TIENES EN LA PANTALLA CUADRADOS Y         TRIANGULOS GRANDES Y PEQUEÑOS"
280 PRINT #1,"  PUEDES INDICAR SI QUIERES PONER ALGO      EN DONDE ESTAN LOS NUMEROS,DICIENDOLO"
290 PRINT #1,"  INDICAME QUE MOVIMIENTO QUIERES"
300 AUX=0
310 LINE INPUT #1,"==>",fr$
320 FR$=UPPER$(FR$)
330 N=1
335 PAL$="ADIOS":GOSUB 2000
336 IF P>0 THEN CLS:END
340 FOR JI=1 TO 12
350 PAL$=MID$(STR$(JI),2,2)
360 GOSUB 2000
370 IF P>0 THEN 400
380 NEXT JI
390 HN=0:GOTO 460
400 HN=-1
410 FOR J=1 TO 12
420 IF P(JI)=PX(J) THEN 450
430 NEXT J
440 GOTO 620
450 PRINT #1,"  NO PUEDES PONER NADA EN EL NUMERO      ";JI;" PORQUE ESTA OCUPADO":GOTO 290
460 PAL$="EN":GOSUB 2000
470 IF P>0 THEN 580
480 PAL$="ENCIMA":GOSUB 2000
490 IF P>0 THEN 580
500 PAL$="SOBRE":GOSUB 2000
510 IF P>0 THEN 580
520 PAL$="DEBAJO":GOSUB 2000
530 IF P>0 THEN 600
540 PAL$="BAJO":GOSUB 2000
550 IF P>0 THEN 600
560 PRINT #1,"  NECESITO QUE ME DIGAS ALGO MAS         PARA SABER EL MOVIMIENTO"
570 GOTO 310
580 AR=-1
590 GOTO 610
600 AR=0
610 PR=P
620 PAL$="CUADRADO":GOSUB 2000
630 IF P>0 THEN 660
640 IF HN THEN  700
650 PRINT #1,"  SI PONES ALGO ENCIMA DE UN TRIANGULO     PUEDE CAERSE.REPITE":GOTO 290
660 IF HN THEN 740 ELSE PB=P
670 N=PB+8
680 GOSUB 2000
690 IF P<>0 THEN PA=PB:PB=P:GOTO  750
700 N=1
710 PAL$="TRIANGULO":GOSUB 2000
720 IF P=0 THEN PRINT #1,"  NO ME DAS SUFICIENTE INFORMACION":GOTO 290
730 PA=P:AUX=6
740 IF HN THEN I=0: P=0:GOTO 860
750 IF PR<PA AND PR<PB THEN CS=2 ELSE CS=1
753 IF NOT AR THEN CS=3-CS
755 IF AUX=0 THEN 758
756 IF (CS=1 AND PB<PA) OR (CS=2 AND PB>PA) THEN 650 ELSE 760
758 IF CS=2 THEN AB=PA:PA=PB:PB=AB
760 FT$=FR$
770 IF CS=2 THEN 810
780 FR$(0)=LEFT$(FT$,PB)
790 FR$(1)=RIGHT$(FT$,LEN(FT$)-PB)
800 GOTO 830
810 FR$(0)=RIGHT$(FT$,LEN(FT$)-PA)
820 FR$(1)=LEFT$(FT$,PA)
830 FOR I=0 TO 1
840 N=1:P=0
850 FR$=FR$(I)
860 WHILE PAL$<>"NO" AND P=0
870 READ PAL$,N(I)
880 GOSUB 2000
890 WEND
900 RESTORE
910 IF P=0 THEN 720
920 IF HN THEN CA=N(0):GOTO 960
930 NEXT I
940 CA=N(0):CB=N(1)
950 IF CA=CB THEN PRINT#1,"  ESE MOVIMIENTO NO LO PUEDES HACER        PORQUE SON DEL MISMO COLOR":GOTO 290
960 PL$(0)="GRANDE":PL$(1)="PEQUENO"
970 IF HN THEN I=0:GOTO 1010
980 FOR I=0 TO 1
990 FR$=FR$(I)
1000 ENC=0
1010 FOR J=0 TO 1
1020 PAL$=PL$(J)
1030 GOSUB 2000
1040 IF P<>0 THEN N(I)=J:ENC=-1
1050 NEXT J
1060 IF NOT ENC THEN 720
1070 IF HN THEN 1090
1080 NEXT I
1090 NA=2*CA-1+N(0)+AUX
1100 IF NOT HN THEN 1130 ELSE NB=0
1110 IF JI MOD 2<NA MOD 2 THEN PRINT #1,"  HAY NO CABE":GOTO 290
1120 GOTO 1200
1130 NB=2*CB-1+N(1)
1170 IF (NB MOD 2)<(NA MOD 2) THEN PRINT #1,"  NO PUEDE HACERSE PORQUE EL DE            ABAJO ES MENOR QUE EL OTRO":GOTO 290
1190 IF OC(NB) THEN PRINT #1,"  ESE CUADRADO YA ESTABA CUBIERTO":GOTO 290
1200 IF NA<7 THEN IF OC(NA) THEN PRINT #1,"  EL CUADRADO DE ABAJO ESTA CUBIERTO":GOTO 290
1210 IF HN THEN 1220 ELSE xx=PY(NB)+20+16*((NB MOD 2)+1)+16*(NA MOD 2):GOTO 1230
1220 IF I MOD 2=0 THEN XX=161 ELSE XX=177
1230 IF XX+T>400 THEN PRINT #1,"  TE SALES DE LA PANTALLA":GOTO 290
1240 I=NA:BR=-1
1250 IF NA>6 THEN GOSUB 3000 ELSE GOSUB 2500
1260 IF HN THEN PX(I)=P(JI) ELSE PX(I)=PX(NB)
1270 IF NOT HN THEN OC(NB)=-1
1280 PY(NA)=XX
1290 BR=0
1300 IF NA>6 THEN GOSUB 3000 ELSE GOSUB 2500
1310 OC(DB(NA))=0
1320 DB(NA)=NB
1330 GOTO 290
2000 REM posicion de fr$ donde esta pal$
2010 M=N
2020 p=INSTR(M,fr$,pal$)
2030 IF p=0 THEN 2120
2040 IF p=1 THEN 2070
2050 A$=MID$(fr$,p-1,1)
2060 IF A$<>" " AND A$<>"," AND A$<>";" THEN 2100
2070 IF p+LEN(pal$)-1=LEN(fr$) THEN 2120
2080 A$=MID$(fr$,p+LEN(pal$),1)
2090 IF A$<>" " AND A$<>"," AND A$<>";" THEN 2100 ELSE 2120
2100 M=p+LEN(pal$)
2110 GOTO 2020
2120 RETURN
2500 REM cuadrado
2510 IF BR THEN P=1:GOTO 2540
2520 p=(i-1)\2
2530 IF p=1 THEN p=3
2540 GRAPHICS PEN p
2550 IF i MOD 2=0 THEN t=16 ELSE t=32
2560 PLOT px(i)-t,py(i)-t
2570 DRAW px(i)+t-1,py(i)-t
2580 DRAW px(i)+t-1,py(i)+t-1
2590 DRAW px(i)-t,py(i)+t-1
2600 DRAW px(i)-t,py(i)-t
2610 MOVE px(i),py(i)
2620 FILL p
2630 RETURN
3000 REM triangulo
3010 IF BR THEN P=1:GOTO 3040
3020 p=(i-7)\2
3030 IF p=1 THEN p=3
3040 GRAPHICS PEN p
3050 IF i MOD 2=0 THEN t=16 ELSE t=32
3060 PLOT px(i)-t,py(i)-t
3070 DRAW px(i)+t-1,py(i)-t
3080 DRAW px(i),py(i)+t-1
3090 DRAW px(i)-t,py(i)-t
3100 MOVE px(i),py(i)
3110 FILL p
3120 RETURN
3500 DATA "AZUL",1,"ROJO",2,"VERDE",3,"NO",0
```



Para que tus dedos no realicen el trabajo duro, M.H. AMSTRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicitanoslo.

Periféricos para AMSTRAD

CONECTAMOS CON TUS IDEAS.



PERIFERICOS PARA LOS MODELOS CPC 464, CPC 664 Y CPC 6128

- ANTA 64 K.3 Ampliación de memoria, buffer de impresora y Ram Disk.
- SINTETIZADOR DE VOZ
  El programa que controla este sintetizador, contiene las reglas básicas de pronunciación en castellano y permite su funcionamiento, tanto en modo directo, como bajo el control de un programa.
- RS-232-C
  Permite comunicar el ordenador con impresoras y plotters con entrada serie, modems, y otros ordenadores.

PERIFERICOS PARA LOS MODELOS PCW 8256 Y PCW 8512

CENTRONICS-RS 232
Proporciona al ordenador dos canales de comunicación:
- Canal paralelo (centronics) para el manejo de impresoras.
- Canal serie (RS-232) para comunicar con otros ordenadores, modems, plotters, etc.).

MHT ingenieros

Distribución: L.S.B., S.A. Sánchez Pacheco, 78. 28002 Madrid.

Para mayor información, escribir a:
L.S.B., S.A. Sánchez Pacheco, 78 - 28002 Madrid
Nombre/Apellidos ....
Domicilio ....
Localidad ....
Provincia ....
C.P. ....
Profesión ....
Edad ....
Información sobre: Periféricos Amstrad ☐ otros ☐

DICCIONARIO DE TERMINOS

La informática es una ciencia que, como todas las demás, emplea su propia terminología, accesible sólo a los iniciados. Los profanos que deseen introducirse a fondo en las entretelas de dicha disciplina no deben verse frenados en su justo empeño durante más tiempo. Aquí está AMSTRAD Semanal para hacerles la vida un poco más fácil. Y si no lo creen, observen detenidamente nuestro diccionario de términos informáticos; el que después de leerlos no entienda cualquier texto o programa del ramo por abstruso que sea, es un azimut, ¿vale?

VERIFYFY: orden del dialecto Basic de la comunidad Gay *«Pompiup* usada para averiguar si su **Amstrad** graba más blanco que el del grupo feminista *«stickdown»*.

COMPUTER: jueves tarde y sábados noche. Se admite Visa. Seriedad y discreción.

TAPE: orden perentoria que, en círculos conservadores, se aplicó intensivamente a las morbideces femeninas exitosamente durante mucho tiempo, pero con el advenimiento de los ordenadores la cosa acabó encinta..., de cassette *(pila de lágrimas)*.

SKIP: el comando de Basic que limpia más blanco.

RANDOM: variante de acceso de ficheros, llamada también aleatoria, protagonista no ha mucho de un famoso film de Silvester Stallone.

RANDOM SPLASH: ficheros aleatorios para hombres curtidos.

MASK: dícese de la marca de chicle preferida por **Amstrad.**

FILTRO: instrucción de un programa que prevé que, si el ordenador se quema durante la ejecución del mismo, el humo que suelte lleve menos nicotina *(ver LIGHT)*.

EDITOR: Maravillosa persona que publica revistas de informática, aceptando trabajos de los lectores, incluso pagándoles.

DIGITAL: proceso consistente en encontrar al astuto asesino por la huella del dedo meñique del pie izquierdo dejada en el interruptor de la luz.

DELETE: Dícese de algo que es propiedad de un extraterrestre.

DECIMAL: programa que siempre apuesta, tras horas de proceso, por el número de la ONCE que nunca toca.

SYNTAX: famoso lingüista griego, mudo todo él. Inventor del bien hablar en la informática *(ver ERROR)*.

ERROR: para empezar, SYNTAX no era mudo. Para terminar, error es un mensaje u orden que da el **Amstrad** durante la ejecución de un programa, pensado para instar al usuario a pronunciar alguna que otra palabra malsonante, como *«caquita»*.

ENSAMBLADOR: persona que junta en un todo coherente las partes de un ordenador. Gracias a ellos no nos dan una caja llena de piezas aisladas al comprar un **Amstrad.**

BYTE: es la bebida preferida por el ordenador.

DIAGRAMA DE FLUJO: dibujo explicativo de un proceso, natural o artificial, que algunas personas se ven obligadas a realizar periódicamente *(ver INCONFESABLE).*

DEBUG: en honor de Jacques Debug, el mayor idiota informático de la historia, autor del programa más largo del mundo capaz de escribir en pantalla *«mi mamá me mima».* El nombre enmarca un proceso destinado a la captura de errores en un programa, o, en su defecto, a la creación de otros nuevos mientras se eliminan los antiguos.

VARIABLE: parte de la memoria de un ordenador a la que se le asigna un valor que nunca coincide con el previsto, alterándose de forma misteriosa *(ver RANDOM).*

TERMINAL: configuración **Amstrad** que incorpora, en lugar de un monitor a color, una ventanilla expedidora de todo tipo de billetes impresos. Incluye, completamente gratis, funcionario *«cutre»,* silla odiosa y cafetería con precios abusivos (más IVA).

SORT: rutina especializada que poseen la mayoría de los sistemas operativos que se precien, pensada para colocar los *«slips»* de los usuarios en orden alfabético.

ROM: barrilito de bebida espirituosa que los **Amstrad** llevan de fábrica. Dicen que es imprescindible para que la máquina funcione *(ver CUBA).*

PROCESADOR: juez militar chileno, apodado *«Z80»,* programado para aplicar la máxima pena aleatoriamente *(ver CRIMINAL).*

RESTORE: del prefijo *«RE»,* repetir, y *«STORE»,* almacenar, es decir, guardar lo que ya está guardado. ¡Vaya estupidez! (Ver *«West Side Store»,* preferiblemente un miércoles).

SILICIO: gobernador romano de California del Siglo I D.J.

SUBRUTINA: trabajo mecánico que se realiza a escondidas y con la desaprobación de todos *(ver HACIENDA).*

MONITOR: especie primate, pequeñita y simpática, que habita encima de las mesas unida por un cable a la unidad central de un ordenador, en lugar de los árboles, como Dios manda. Originario de Sudáfrica, sólo existe en dos colores: verde y multicolor.

NETWORK: ciudad norteamericana en la cual todos sus habitantes trabajan juntos conectados por cables, compartiendo calles, coches y edificios por riguroso orden.

REGISTRO: subdivisión de un fichero armada de orden judicial, que normalmente acaba revolviendo los cajones *(con perdón)* de la morada de algún probo usuario.

CHAIN: país asiático especializado en el encadenamiento de programas, llamados ciudadanos por las malas lenguas.

FICHERO: palo inmaterial, compuesto de células de memoria y acabado en un gancho, que se utiliza en las faenas de pesca de unicornios.

Empleado provisto de tarjeta perforada por un reloj.

Bueno, pues aquí acaba, por ahora, el extracto de lo más florido de nuestro diccionario informático. Tiene algunas pequeñas deficiencias: está incompleto, no está ordenado alfabéticamente y hace referencias a palabras que no existen en el texto, pero por lo demás es una joya. En caso de duda, no lo dude: no nos llame, nosotros tampoco le llamaremos.



Ofites

Presenta: el universo del software, y

DELTA +

La más moderna base de datos DELTA, superándose a sí misma, "DELTA +", desarrollada para CP/M por COMPSOFT con todo en español.

Diseña sus propios ficheros; desde un simple fichero de nombres y direcciones hasta su propio sistema contable. El formato standar DIF permite intercambiar datos en DELTA, desde las hojas de cálculo CRACKER II, etc... y viceversa. Intercambio de datos con la mayoría de los tratamientos de texto como NEWWORD para MAILING.

Incluye un sencillo y funcional sistema de impresión de etiquetas con: hasta 5 columnas de etiquetas, 65 caracteres por etiquetas, 20 líneas con 3 campos cada una.

- PROGRAMABLE Y RELACIONAL.
- FICHEROS INDEXADOS.
- HASTA 90 CAMPOS ó 2.000 CARACTERES.
- MULTIPLES SISTEMAS DE BUSQUEDA, 8 CLAVES.
- FICHEROS DE HASTA 8 Mb.
- 8 GRUPOS DE TRANSACCION POR REGISTRO.

BASE DE DATOS

17.850 pts.

NEWWORD

Programa de tratamiento de textos mejorando todo lo anterior. Manual y programa en español, que le enseñarán con facilidad y rapidez lo más avanzado en procesadores de textos. Compatibilidad funcional con WORDSTAR incluyendo muchas capacidades adicionales.

Tiene un potente MAIL-MERGE con opción de selección de destinatarios por criterios base de datos, creación de documentos, impresión de etiquetas. Utiliza todo el espacio de disco. Ensamblaje de textos, sustitución, etc., de la forma más fácil: autohace copias de seguridad. ¡NUNCA PERDERA UN TEXTO!

- Ñ, ACENTOS, DIERESIS, ETC...
- PRESENTACION EXACTA EN PANTALLA DEL FUTURO DOCUMENTO IMPRESO.
- INTERCAMBIOS DE FICHEROS CON CRACKER.
- VARIABLES SUSTITUIBLES EN IMPRESORA.
- POTENTE CALCULADORA.
- COMPROBADOR ORTOGRAFICO Y GRAN DICCIONARIO (45.000 TERMINOS AMPLIABLES).
- POSIBILIDAD DE LECTURA DE FICHEROS DE DELTA, CARD BOX, SUPERCALC, DBASE II, ETC...

TRATAMIENTO DE TEXTOS

17.850 pts.

CRACKER II

El CRACK de las hojas de cálculo, la que deja detrás al resto. Funciones nunca vistas, formateo de fechas, salvaguardia continua sobre un fichero. Realiza automáticamente copias de seguridad. Además de las tradicionales funciones, CRACKER II posee funciones lógicas, estadísticas y de alta matemática. Intercambia datos con NEWWORD, bases de datos y la mayoría de las hojas de cálculo.

- CELDAS PROGRAMABLES.
- FUNCIONES ESPECIALES: Fecha, días; desde y hasta la fecha de la semana, del año, lapso de tiempo, retraso, beep entrada, saludo usuario.
- SISTEMA DE AYUDA ON-LINE.
- SUMA CONDICIONAL.
- TOMAR DECISIONES EN LA HOJA.
- 18 MODOS GRAFICOS DISTINTOS.
- TRADICIONALES FUNCIONES MATEMATICAS Y AMPLIACION, FUNCIONES ESTADISTICAS Y LOGICAS.
- GENERA GRAFICOS EN BASE A LOS DATOS.

HOJA DE CALCULO

17.850 pts.

EDITOR Y DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA





HISOFT-C

Daniel Palomo Ortega

Todos los usuarios de Amstrad estábamos deseando que apareciese en España un compilador de C para nuestro ordenador. Por fin vemos nuestros deseos cumplidos, con un magnífico compilador como es el de HISOFT. MICROHOBBY AMSTRAD ha probado este compilador, en versión inglesa, para daros una visión del funcionamiento del mismo.



El lenguaje C se creó en 1972 como una herramienta de programación. El creador del C es Dennis Ritchie, de los laboratorios Bell. Este lenguaje surgió cuando Ritchie trabajaba, junto con Ken Thompson, en el diseño del sistema operativo UNIX.

Este lenguaje no surgió por arte de magia, como es de suponer, sino que derivó del lenguaje B de Thompson, que a su vez... Pero eso es otra historia.

Fue creado como herramienta de programadores, por tanto es un lenguaje útil, ya que es potente, flexible y rápido.

El C es un lenguaje muy extendido, existen compiladores para multitud de sistemas aparte del UNIX, tan importantes como: Cray I, IBM, Sperry, Apple, Commodore, Amstrad, etc.

El C se utiliza en aplicaciones muy variadas, por ejemplo, el sistema operativo UNIX está escrito en C. Muchos compiladores, además de multitud de juegos, se han hecho con este lenguaje e incluso se utilizó en *El retorno del Jedi* para la animación de las secuencias de la película. Paquetes de software, programas de gestión y un largo etcétera han sido realizados en C.

El C es un lenguaje estructurado que produce programas compactos, eficientes, transportables, y de modificación muy sencilla.

¿Qué es lo que da esta potencia? C es un lenguaje moderno que trata las tareas por separado, es decir, es un lenguaje modular, todo esto se realiza gracias a la flexibilidad de sus expresiones. La filosofía de diseño de C se basa en el adecuado uso de las funciones, éstas son parecidas a las subrutinas, procedures y funciones de otros lenguajes, el C trabaja siempre con funciones como palabras clave, **PRINTF(), SCANF(), FSEEK(),** son funciones predefinidas, y nosotros podemos crear nuestra biblioteca muy fácilmente e incluir funciones propias en las ya creadas. Por todo esto, muchas de las funciones a las que estamos acostumbrados no están implementadas en C.

La implementación de HISOFT es buena y además tiene el detalle de suministrar dos compiladores en el mismo disco, uno que funciona bajo AMSDOS y otro bajo CPM. Pasemos a analizarlos.

Crear un programa fuente

Lo primero que tenemos que hacer para crear un programa C es almacenarlo en un fichero de texto, esto lo podemos realizar con los editores que se suministran con cada compilador.

El perteneciente a **AMSDOS** es el típico de Hisoft con órdenes parecidas a las de anteriores productos de esta marca.

El de CP/M es, el ya conocido por sus usuarios, ED80 compatible con WORDSTAR.

Podemos utilizar cualquier editor al que estemos más acostumbrados.

Veamos cómo funcionan.

El editor de líneas

El editor que va implantado en el compilador de AMSDOS es muy parecido al original de **Amstrad,** teniendo habilitado el cursor de copia y todas las demás funciones.

Los números de línea son una referencia para el programador y el compilaor sólo los tiene en cuenta en el tratamiento de errores, para referenciarnos más rápidamente a la línea en cuestión. Si en este momento se pulsa la tecla [E] se editará la línea que produjo el error, cualquier otra tecla retorna al modo directo. Las llaves son imprescindibles en C, con ellas se indica el principio ({)y el final (}) de funciones, condiciones, bucles, etc. En el CPC6128 se pueden obtener con [CTRL] más los paréntesis.

COMANDOS DEL E

In,i:	El editor imprime los números de línea siendo n el número inicial e i el incremento.
Ln,m:	Lista un número indicado de líneas, siendo n la primera línea a listar y m la última.
Wn,m:	Lista por impresora.
S,,d:	Selecciona el delimitador a utilizar por los comandos del editor, muy útil cuando queremos buscar cadenas que contienen comas, d es el delimitador elegido, éste no puede ser un espacio.
C:	Retorna al compilador. Si todo es correcto tecleando #INCLUDE comenzará a compilarse nuestro programa.
Dn,m:	Borra los números de línea comprendidos entre n y m.

Los RSX de AMSDOS están todos a nu



DITOR DE LINEAS

Nn,i:	Renumera el programa siendo n el primer número de línea e i el incremento.
Fn,m,c,c:	Busca una cadena c en los números de línea comprendidos entre n y m, y la sustituye por la cadena c.
V:	Imprime en pantalla los valores actuales de n, m, c y c, así como el delimitador actual.
Pn,m, nom:	Graba un programa en disco o cinta, n es la primera línea que se quiere grabar, m la última y nom el nombre del fichero con su extensión.
Gn,m, nom:	Carga un programa de disco o cinta.
En:	Edita la línea especificada por n.

əstra disposición.

El editor de CP/M + (ED80)

Este editor, compatible con WORDSTAR, es muy potente y no tiene nada que envidiar a los procesadores comerciales.

Todos los comandos se consiguen con [CTRL] y una o varias letras.

A continuación damos una lista de ellos clasificados por tareas:

Movimiento del cursor	
Ŝ Carácter izq.	D̂ Carácter der.
Ĥ Caràcter izq.(bor)	
Â Palabra izq.	F̂ palabra der.
Ô S Tabula izq.	Ô D Tabula der.
Q̂ S Principio línea	Q̂ D Final línea
Ê Línea superior	X̂ Línea inferior
Ô E Principio texto	Ô X final de texto
R̂ Página posterior	Ĉ Página anterior

Borrado	
Ŷ	Borra línea
[DEL] Borra último carácter	Ĝ Borra este carácter
Ô T Borra palbr. izq.	T̂ Borra palbr. der.
Q̂ [DEL] Borra principio línea	Q̂ Y Borra final de línea
K̂ B Marca princ. bloque	K̂ K Marca fin bloque
K̂ V Mueve bloque	K̂ C Copia bloque
K̂ Y Borra bloque	K̂ R Lee bloque de disco
K̂ W Graba bloque disco	

Movimiento rápido del cursor	
Ô G Ir a línea	
Q̂ B Ir a princ. de bloque	Q̂ K Ir a fin de bloque
K̂ O Recuerda posición	Q̂ O Ir a posición

Búsqueda y sustitución	
Q̂ F Busca primero	L̂ Busca siguiente
Ô L Sustituye y busca	Ô A Sustituye todo
K̂ Q Abandona y sale	Ô Q Salir sin Backup
K̂ X Salir con Backup	



Varios	
K̂ F Directorio	K̂ J Borra fichero de disco
P̂ Control meta-key	Ĵ Ayuda

Un editor muy completo, como se puede ver, y con posibilidades de adaptarlo a nuestro gusto, permitiéndonos remodelar la pantalla de acuerdo al monitor utilizado, ya sea de la serie CPC o PCW.

Un poco de... C

El C es el lenguaje que se maneja muy bien. En eso de los números el rango es el mismo que el de LOCOMOTIVE BASIC, pero hay algunas aclaraciones que hacer; los números octales se representan añadiendo un 0 a la izquierda si tiene significado, por lo tanto, no debemos utilizar esta notación, ya que sería entendido erróneamente por el compilador. Los números hexadecimales se representan con un cero seguido por una X y a continuación el número hexadecimal *(OX4F3B)*.

Los caracteres simples pueden ser manejados encerrándolos entre comillas simples o dobles *(**'A';''B''**)*.

Tenemos una serie de caracteres, llamados en C caracteres de escape, que son utilizados para imprimir retorno de carro, salto de página, comillas, etc. Estos se representan precedidos de la barra atrás (\) y son los siguientes:

Es la animación d\n nueva línea
\b espacio atrás
\f avance de página
\t tabulador horizontal
\r retorno de carro
\'' comillas
\\ barra atrás
\'' comillas simples

También podemos utilizarlos escribiendo su valor ASCII en octal precedido por la barra atrás, esto sirve para imprimir todos los caracteres gráficos de **Amstrad** por ejemplo \32 sería EOF; \377 imprimiría el último carácter del juego (las pesas), el valor a continuación de la barra atrás únicamente se puede poner en las bases disponibles.

Las cadenas se almacenan en una matriz conteniendo el valor ASCII de cada componente de la misma, cada carácter ocupa un byte.

Los nombres e identificadores pueden tener la longitud que deseamos, para hacer más claro lo que queremos representar con ellos, pero sólo tendran significado para el compilador, los 8 primeros caracteres del nombre, se pueden utilizar todas las letras, números y el guión abajo, (A-Z, a-z, 0-9, _).

Por convenio se dejan las palabras mayúsculas para los nombres que utilicemos en los comandos del preprocesado, se escribe el programa, siempre que sea posible, en minúsculas.

Palabras reservadas

C es mucho menos extenso, en lo que a palabras reservadas se refiere, ya que al estar compuesto por funciones es más difícil, por no decir imposible, utilizar un nombre de una función como nombre de variable, ya que los paréntesis forman parte del nombre de la función y no son caracteres permitidos para un nombre de variable.

Pero sí tenemos algunas palabras con las que el compilador se haría un lío si las utilizáramos como nombre de variable.

Estas palabras son las siguientes:

PALABRAS RESERVADAS

Auto	break
char	continue
double	else
float	for
inline	int
return	short
struct	switch
unsigned	while
case	cast
default	do
entry	extern
goto	if
long	register
sizeof	static
typedef	union

Programas en C

Los programas en C se consiguen gracias a una colección de funciones creadas por nosotros, las implementas o las de librería, que hacen que todo eso funcione. ¿Cómo consigue saber el compilador por donde ha de empezar a ejecutar el programa? Esto lo sabe porque nosotros hemos de llamar, obligatoriamente, Main () a la función principal, esta es la traducción de Main, de nuestro programa, así sabrá por dónde empezar, sencillo, ¿no?

A continuación del nombre de la función se pueden poner los parámetros de la misma, encerrada entre paréntesis vacíos a continuación del nombre. La función comienza en el punto en que encuentra una llave de apertura ({), todo lo que existe entre el nombre y esta llave es considerado por el compilador como declaración de variables. La sintaxis de una función es la siguiente:

```
Nombre de función (lista de
parámetros)
Declaración de variable
'
Asignación de variable
y llamadas a funciones
'
```

La llave de cierre finaliza la función. Los comentarios pueden ser incluidos entre barras y asteriscos (/* comentario*/)

Tablas de operadores

A continuación se da la tabla de operadores, ordenados, de acuerdo con su prioridad de mayor a menor:

Operador Prioridad

()	16	Paréntesis
func()	15	Llamada a una función.
[]	15	Subíndice.
.	15	Suministro del valor de un incremento de una estructura.
→	15	Puntero a un miembro de una estructura.
*	14	Operador de indirección
&	14	Dirección de var.
—	14	Signo menos.
!	14	NOT negación lógica.
++	14	Operador de incremento.
——	14	Operador decremento.
sireof	14	Suministra el número de bytes ocupados por una variable.
cast	14	Fuerza a que el tipo sea el especificado.
*	13	Multiplicación.
/	13	División.
%	13	Halla el resto de una división.
+	12	Suma.
—	12	Resta.
> >	11	Operador de shift+ a la derecha.
< <		Operador de shift+ a la izquierda.
<		
<	10	Operadores de relación.
< =		
> =		
= =	9	Operador de igualdad.
!=	9	Operador de no igualdad.
&	8	Y lógico bit a bit.
ˆ	7	XOR lógico bit a bit.
|	6	O lógico bit a bit.
&&	5	Y lógico.
||	4	O lógico.
?:	3	Operador condicional.

Tipos de datos

En C hemos de declarar todas las variables que vayamos a utilizar en el programa, tal como ocurre en otros lenguajes compilados; para ello tenemos algunos tipos ya implementados y éstos son:

CHAR para declarar variables y matrices en cadena
INT para declarar enteros con signo
LINSIGNED para declarar enteros sin signo
FLOAT para declarar enteros variables en punto flotante (reales)
SHORT para declarar enteros cortos (1 byte)
LONG para declarar enteros largos (2 bytes)
TYPEDEL para definir nuestros propios tipos de datos
* para declarar punteros

Las matrices en C son muy parecidas a las de BASIC. Esto se hace con el identificador de tipo correspondiente, seguido del nombre de la variable y los subíndices entre corchetes.

CHAR nombre [10] [30]

Si no se usa ningún subíndice, se utiliza el espacio requerido por los elementos de la matriz.

Las estructuras es algo que posee C y que da gran flexibilidad a la hora de hacer programas, ya que nos permiten referirnos a cualquiera de los elementos de la estructura muy fácilmente, consiguiendo crear un base de datos, por ejemplo, sin problemas, ya que en la estructura se puede incluir cualquier tipo de datos.



Banco de PRUEBAS

Los modos de almacenamiento determinan qué variable conoce cada función y hasta cuándo deben permanecer en memoria estas variables.

Los tipos de almacenamiento son:

AUTO: Es el modo asumido por defecto, estas variables tienen alcance local, es decir, sólo las conoce la función en la cual se declaran estas variables, desapareciendo al finalizar la misma.

EXTERN: Cuando una variable se declara fuera del cuerpo de una función se dice que es externa y estas variables pueden ser utilizadas por todas las funciones que declaren en su interior como «extern», incluso estando esta variable en otro módulo que queramos complilar junto con el que utiliza esa variable.

Si no se declara como **«extern»** se considera una variable nueva y asumirá el modo por defecto **(auto)**.

STATIC: Es del mismo alcance que auto, local, pero permanece en memoria durante toda la ejecución del programa.

REGISTER: Estas variables son de modo automático, pero el compilador intenterá que estén situadas en los registros de la CPU en vez de estar en la memoria RAM. Es más una súplica que una orden.

El preprocesador

El prepocesador es una parte del compilador C, se encarga de realizar algunas tareas «domésticas», tales como definir constantes y macros, incluir ficheros y librerías etc. Los comandos son los siguientes:

#define: Define una constante, una macro instrucción, o cualquiera de las palabras reservadas pudiéndose utilizar el nombre impuesto por nosotros en su lugar. Por ejemplo:
#define EOF —1

#error: Este comando retira de memoria el texto de los códigos quedando como única diferencia los números de error. Esto es útil cuando tenemos que compilar programas muy largos ya que el texto de errores ocupa 2K

#data: Se sitúa al principio de la zona de datos, la dirección de memoria debe ir en hexadecimal, sólo utilizable en CPM.

#list: Activa(+) o desactiva(—) el listado del programa.

#direct: Activa o desactiva la ejecución directa de comandos, parecido a la ejecución directa en basic.

#include: Incluye el fichero escrito a continuación ya sea para compilar juntos los módulos o para incluir funciones de biblioteca. Tiene dos modos posibles:

1) #include **«Nomfich»**
   #include nomfich

incluye todo el fichero compilador junto con el actual.

2) #include ?nomfich?

incluye sólo funciones utilizadas por el programa que da la orden **#include**.

#translate Graba en disco el código objeto con el nombre especificado por nosotros en CP/M sólo sirve para cambiar el nombre del fichero objeto.

Las bibliotecas de programa

El compilador de C incluye varias bibliotecas, con las que podremos cubrir casi todas nuestras necesidades. Estas bibliotecas contienen todas las funciones típicas de C, las rutinas de manejo del disco, etc. Tenemos dos bibliotecas con casi todas las palabras reservadas de Basic pero al estilo C, con ellas se pueden aprovechar todas las prestaciones del AMTRAD tanto gráficas como de sonido e interrupciones.

En CP/M encontramos todas las funciones para manejar a nuestro antojo, tanto el disco como pantalla o teclado. Los usuarios de CP/M+ podrán disfrutar de los gráficos, ya que incluye una biblioteca para la extensión gráfica GSX.

Las bibliotecas se dividen en dos partes:

El encabezamiento que es donde se definen todas las constantes que se utilizan en la correspondiente biblioteca.

Y la biblioteca en sí, que es donde se encuentran todas las funciones utilizables y algunas para el manejo interno.

Hay que agradecer a Hisoft que sea tan explicativo en estas bibliotecas, dando siempre la oportunidad de aprender algo más, ya que éstas se encuentran llenas de comentarios y nombres de variables con un significado muy claro, que hacen muy fácil su lectura y comprensión.

En las figuras 1 y 2 se encuentran las listas de funciones de las bibliotecas de AMSDOS y CP/M, stdio.h es la misma para los compiladores.

STUDIO.LIB

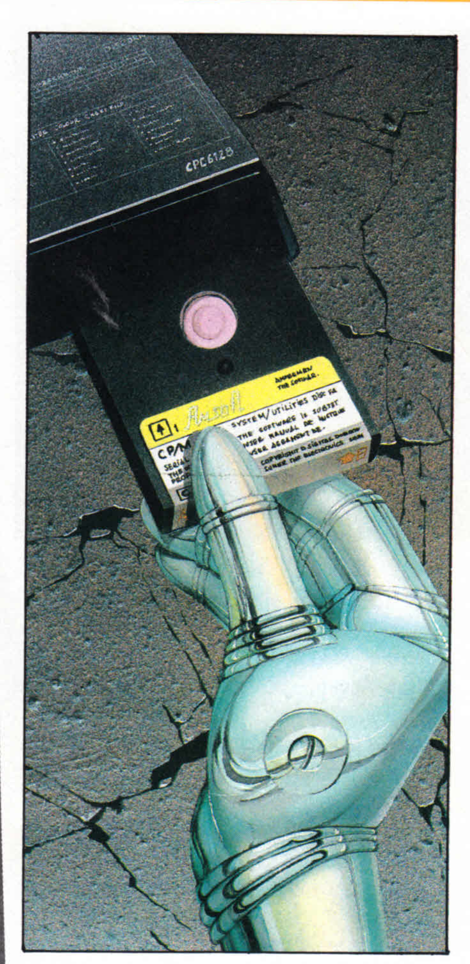
```
int abs(n)
int sgn(n)
int poek (adress)
void poke (adress,value)
in out(data, port)
int inp(port)
iont atoi(s)
void qsort(list, nu-,items, size, cmp-func)
char *strcat(base, add)
char *strncat(base,add,number)
int strcmp(s,t)
int strncmp(s1, s2, n)
char *strcpy(s1, s2, n)
char *strcmp(dest, source)
unsigned strlen(s)
char *strchr(string, ch)
int strspn(s1, s2)
int strcspn(s1, s2)
char *strchr(s, c)
char *strrchr(s, c)
int ispunct(c)
int isalnum(c)
int isxdigit(c)
int isascii(c)
int iscntrl(c)
int isprint(c)
int isgraph(c)
int toascii(c)
char *fgets(s, n, fp)
char *gets(s)
void fputs(s, fp)
char *calloc(n, size)
void free(block)
char *sbrk(n)
exit(n)
-exit(n)
void srand(n)
void long-multiply(c, a, b)
void long-init(a, n1, no)
void long-set(a, n, d)
void long-copy(c, a)
```

BASIC1.LIB SONIDOS

setup-sound()
play(control-string,status)
S-arm-event(channel-bit,seb-add)
S-queue(sp)
sound-check(chanbit)
S-release(channel-bits)
S-ampl-envelope(number, envelope)
S-tone-envelope(number, envelope)
S-hold()
S-CONTINUE()
after(delay-in-ticks, control-block, function-name)
every(period-in-ticks, control-block, function-name)
add-ticker (ctrl-block, initial-time-delay, recharge-delai, function-name
init-event (event-block, function-name)
border(colour)
cass-speed (speed)
catalog()
cls()
event-enable()
event-disable()
flash-speed(time1, time2)
ink(ink-to-setup, colour1, colour2)
int inkey(key-number)
char *instr(main-string, sub-string)
itob(n, string, precision)
joy(joystick-number)
int key-function(traslated-key-number, expansion-string)
K-arm-breaks(event-routine, ROM-select)
K-disarm-break()
read-file(filename, adress)
char *strlower(string)
char *strupper(string)
time(array)

BASIC2LIB

draw(control-string)
T-set-graphic(on)
T-swap-streams(stream-number, another-stream-number)
T-get-cursor(px-column, py-row, p-roll-count)
G-ask-cursor(pdx, pdy)
G-set-origin(x,y)
G-win-width(x1,x2)
G-win-height(y1,y2)
G-clear-window()
G-set-pen(ink)
G-set-paper(ink)
G-wr-char(c)
G-move-absolute(x,y)
G-move-relative(dx, dy)
G-plot-absolute(x, y)
int G-test-absolute(x, y)
int G-test-relative(dx, dy)
G-lie-absolute(x, y)
G-line-relative(dx, dy)



CPM.LIB

void cpm-cmd-line(aargc, aargv, buffer)
int parse-args(s, argv, dest)
cpm-dir(drive, user, afn, sp, width)
void fcb-to-name(filename, fcb)
cpm3-bios(func, a-param, bc-param, de-param, hl-param)
int cpm-punout(c)
int cpm-version()
int unlink(filename)
rename(oldname, newname)
int cpm-user(new-user)
void cpm-bdos(func, param)
void fseek(stream, offset, mode)
read-file(filename, address)
char *strupper(string)
char *strlower(string)
write-file(filename, address, length)

GSX.LIB gráficos de CP/M

v-clswk(handle)
v-updwk(handle)
v-pline(handle,count,pxyarray)
v-pmarker(handle,count, pxyarray)
v-gtext(handle,x,y,string)
v-bar(handle,pxyarray)
vst-height(handle,heigth,char-width, char-height,cell-width,cell-height)
int vst-rotation(handle,angle)
int vsl-type(handle,style)
int vsm-type(handle,symbol)
int vsf-color(handle, color-index)
int vsm-locator(handle,x,y,xout,yout,term)
vsin-mode(handle,dev-type,mode)
v-exit-cur(handle)
v-enter-cur(handle)
v-curright(handle)
v-curhome(handle)
v-eeol(handle)
v-rvon(handle)
v-dspcur(handle,x,y)
v-rmcur(handle)
gsx()

Conclusiones

Este es un compilador fácil de usar en sus dos versiones, trae en sus bibliotecas multitud de funciones con las que podremos hacer casi todo, lo que no podamos hacer sólo tenemos que implementarlo.

Se probó su rapidez con dos programas, 1 y 2, demostrando esacasa diferencia entre el compilador de AMSDOS y el de CP/M. Las pruebas efectuadas dieron los siguientes resultados:

Compilación	3 segundos
Bucle vacío	2 segundos en 1.000 vueltas
Bucle simple	6 segundos en 500 vueltas
Bucle complejo	49 segundos en 500 vueltas
Bucle texto	18 segundos en 100 vueltas

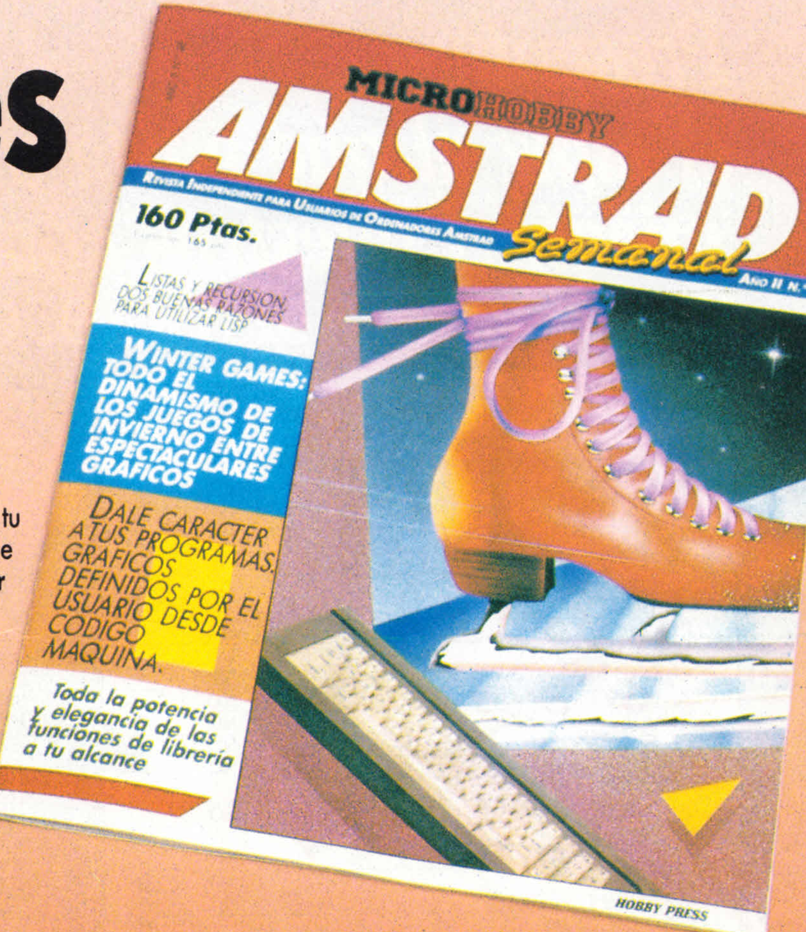
Podéis probar a ejecutar los mismos bucles en basic, midiendo los tiempos tardados, y haciendo las oportunas comparaciones. Pero por si era poco miramos también cómo maneja C los ficheros. El programa 2 leía un fichero de texto, concretamente stdio.h, con más de 2.000 caracteres en tan sólo 4 seg., el procesador de textos con que se ha escrito este artículo, tarda el mismo tiempo en realizar esa operación, a pesar de estar escrito en puro ensamblador .

Los manuales son escuetos pero bastante completos, tratando cada punto lo justo, sin pasarse en ningún sentido. La única pega encontrada fue la de estar escritos en inglés, a ver si tenemos pronto la versión española.

EQUIPO: CPC 464-664-6128 PCW 8256-8512
SISTEMA OPERATIVO: AMSDOS Y CP/M
DISTRIBUIDOR:
OFITES INFORMATICA
PRECIO: 15.000 PTAS.

OFERTA ESPECIAL I ANIVERSARIO

6 meses Gratis
de AMSTRAD CASSETTE



Suscríbete ahora a Microhobby Amstrad, o realiza tu renovación, y recibirás, totalmente gratis, un regalo de excepción: **una suscripción a Amstrad Cassette por seis meses.**

Cada cinta contiene los programas publicados por Microhobby Amstrad durante un mes. **Todos los programas de nuestras cintas se encuentran desprotegidos, con el objeto de facilitar su copia en disco y la revisión de los listados.**



En cada cinta encontrarás:

— Apasionantes juegos llenos de acción y dinamismo.
— Utilidades con las que sacar mayor partido a tu ordenador.
— Rutinas en código máquina, para que las utilices en tus propios programas.
— Y pequeños trucos de programación, para que, poco a poco, te conviertas en un experto.

Recorta o copia el cupón que aparece cosido en las páginas de esta revista.
APROVECHA ESTA OFERTA UNICA válida sólo para España hasta el 31 de noviembre de 1986.

Ofites Informática

Presenta:
el lápiz al que gusta decir SI mientras nuestros competidores dicen no

UNICO PARA AMSTRAD, CON PRECISION PIXEL

FUNCIONES	ESP	dk'tronics	OTROS
UNICO MENU DE PANTALLA	SI	NO	
ARRASTRE OBJETOS PANTALLA	SI	NO	
TRASLADO OBJETOS PANTALLA	SI	NO	
TRASLADO DE CURSOR	SI	NO	
CAJAS ELASTICAS	SI	SI	
LINEA ELASTICA	SI	SI	
TRIANGULO ELASTICO	SI	NO	
ELIPSE ELASTICO	SI	NO	
DIAMANTE ELASTICO	SI	NO	
POLIGONO ELASTICO	SI	NO	
HEXAGONO ELASTICO	SI	NO	
OCTOGONO ELASTICO	SI	NO	
CUBO ELASTICO	SI	NO	
PIRAMIDE ELASTICA	SI	NO	
CIRCUNFERENCIAS	SI	SI	
CIRCULOS RELLENOS	SI	NO	Compare con otros lápices
CAJAS RELLENAS	SI	NO	
ELIPSES RELLENAS	SI	NO	
CUÑAS	SI	NO	
SIMULADOR DE CORTES	SI	NO	
DISEÑO DE ZOOM	SI	SI	
IMAGEN ESPEJO E INVERTIDA	SI	NO	
FONDO DE REFERENCIA	SI	NO	
REJILLA DE FONDO	SI	NO	
OPCION DISPLAY X, Y	SI	NO	
RELLENADO CON COLOR	SI	SI	
LAVADO DE COLOR	SI	NO	
VOLCADO PANTALLA RESIDENTE	SI	NO	
DIBUJO DE BORDES EN 3 D	SI	NO	
TEXTO	SI	SI	
9 TAMAÑOS DE BROCHA	SI	NO	
18 TOBERAS MOSTRADORAS	SI	NO	
4 MEZCLAS BASICAS	SI	NO	
VARIADOR DE MEZCLAS	SI	NO	
SOMBREADO DE MEZCLAS XOR	SI	NO	
FICHERO ICONOS RESIDENTES	SI	NO	
FICHERO RELLENOS RESIDENTES	SI	NO	
26 COLORES DE PAPEL	SI	NO	
PALETA DE 15 TONOS DE COLOR	SI	NO	
POSICIONAMIENTO DE PUNTO	SI	SI	
RAYOS DESDE UN PUNTO FIJO	SI	NO	
DIBUJO REFLEJADO (ESPEJO)	SI	NO	
FUNCION HOME	SI	NO	
CONTROL DESDE TECLADO	SI	SI	
CONTROL CON JOYSTICK	SI	NO	
DISPONIBLES MODOS 1 Y 2	SI	?	

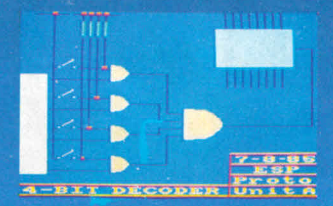
DEBIDO A LA FALTA DE ESPACIO NO PODEMOS LISTAR LAS OTRAS 40 FUNCIONES MAS QUE NUESTRO LAPIZ ES CAPAZ DE HACER.

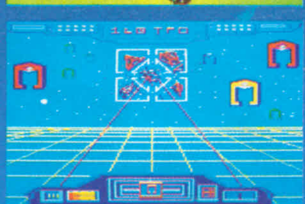


ESTOS SON ALGUNOS EJEMPLOS DE LOS GRAFICOS QUE VD. PODRA REALIZAR CON NUESTRO LAPIZ OPTICO









DE VENTA EN LOS MEJORES COMERCIOS DE INFORMATICA

Si Vd. tiene alguna dificultad para obtener el lápiz óptico, puede dirigirse a:



Avda. Isabel II, 16 -8º
Tels. 455544 - 455533
Télex 36698
20011 SAN SEBASTIAN

DISPONIBLE PARA:

CPC 464 CASSETTE .... 4.900 Ptas.
CPC 464-664 DISCO .... 6.900 Ptas.
CPC 6128 DISCO .... 6.900 Ptas.

(IVA no incluido)

CONDICIONES ESPECIALES PARA DISTRIBUIDORES

Versión SPECTRUM, AMSTRAD y COMMODORE

Alístate a Juegos & ESTRATEGIA

LA BATALLA DE INGLATERRA ha comenzado



Todas las unidades de la RAF están bajo tu mando, y la Lutwaffe —tu ordenador— intentará neutralizarlas. El destino del mundo libre depende de ti.

Oferta especial hasta el 31 de noviembre: PIDE TRES NUMEROS Y PAGA SOLO DOS.

ENVIE HOY MISMO ESTE CUPON AL APARTADO 232 DE ALCOBENDAS (Madrid)

☐ Deseo recibir en mi domicilio tres ejemplares de **Juegos & Estrategia al precio especial de 2.255 ptas.,** lo que me supone adquirir tres y **pagar sólo dos.** Marco los tres ejemplares que deseo con una cruz.

☐ Deseo recibir un solo ejemplar de **Juegos & Estrategia** al precio de 1.125 ptas. Marco con una cruz el ejemplar que deseo recibir.

Spectrum	**Amstrad**	**Commodore**
N.º 1 ☐ Arnhem	☐ Arnhem	
N.º 2 ☐ Ratas del Desierto	☐ Ratas del Desierto	
N.º 3 ☐ OTAN Alerta War Zone	☐ Teatro de Europa War Zone	☐ Teatro de Europa
Especial 1 ☐ Elecciones Generales		
N.º 4 ☐ Su mejor hora (La batalla de Inglaterra)	☐ La batalla de Inglaterra	☐ La batalla de Inglaterra

NOMBRE .... Fecha de nacimiento ....

DIRECCION ....

LOCALIDAD .... PROVINCIA ....

C. POSTAL .... TELEFONO .... PROFESION ....

Forma de pago:

☐ Talón bancario a nombre de Hobby Press, S. A. ☐ Giro Postal a nombre de Hobby Press, S. A., n.º de giro ....

☐ Tarjeta de crédito: Visa n.º .... Master Charge n.º .... American Express n.º ....

Fecha de caducidad de la tarjeta .... Fecha y firma ....